Semi-Algebraic Off-Line Range Searching and Biclique Partitions in the Plane

Pankaj K. Agarwal
Department of Computer Science, Duke University, Durham, NC, USA

Esther Ezra
Department of Computer Science, Bar-Ilan University, Ramat-Gan, Israel

Micha Sharir
School of Computer Science, Tel Aviv University, Tel Aviv, Israel

Abstract

Let P be a set of m points in $\mathbb{R}^2$, let Σ be a set of n semi-algebraic sets of constant complexity in $\mathbb{R}^2$, let $(S,+)$ be a semigroup, and let $w : P \to S$ be a weight function on the points of P. We describe a randomized algorithm for computing $w(P \cap \sigma)$ for every $\sigma \in \Sigma$ in overall expected time $O^*\big(m^{\frac{2s}{5s-4}} n^{\frac{5s-6}{5s-4}} + m^{2/3}n^{2/3} + m + n\big)$, where $s > 0$ is a constant that bounds the maximum complexity of the regions of Σ, and where the $O^*(\cdot)$ notation hides subpolynomial factors. For $s \geq 3$, surprisingly, this bound is smaller than the best-known bound for answering m such queries in an on-line manner. The latter takes $O^*(m^{\frac{s}{2s-1}} n^{\frac{2s-2}{2s-1}} + m + n)$ time.

Let $\Phi : \Sigma \times P \to \{0,1\}$ be the Boolean predicate (of constant complexity) such that $\Phi(\sigma,p) = 1$ if $p \in \sigma$ and 0 otherwise, and let $\Sigma\,\Phi\,P = \{(\sigma,p) \in \Sigma \times P \mid \Phi(\sigma,p) = 1\}$. Our algorithm actually computes a partition $\mathcal{B}_\Phi$ of $\Sigma\,\Phi\,P$ into bipartite cliques (bicliques) of size (i.e., sum of the sizes of the vertex sets of its bicliques) $O^*\big(m^{\frac{2s}{5s-4}} n^{\frac{5s-6}{5s-4}} + m^{2/3}n^{2/3} + m + n\big)$. It is straightforward to compute $w(P \cap \sigma)$ for all $\sigma \in \Sigma$ from $\mathcal{B}_\Phi$. Similarly, if $\eta : \Sigma \to S$ is a weight function on the regions of Σ, $\sum_{\sigma \in \Sigma : p \in \sigma} \eta(\sigma)$, for every point $p \in P$, can be computed from $\mathcal{B}_\Phi$ in a straightforward manner. We also mention a few other applications of computing $\mathcal{B}_\Phi$.

2012 ACM Subject Classification Theory of computation

Keywords and phrases Range-searching, semi-algebraic sets, pseudo-lines, duality, geometric cuttings

Digital Object Identifier 10.4230/LIPIcs.SoCG.2024.4

Related Version *Full Version*: `https://arxiv.org/abs/2403.12276` [11]

Funding *Pankaj K. Agarwal*: Partially supported by NSF grants CCF-20-07556 and CCF-22-23870 and by a US-Israel Binational Science Foundation Grant 2022131.
Esther Ezra: Partially supported by Israel Science Foundation Grant 800/22 and US-Israel Binational Science Foundation Grant 2022131.
Micha Sharir: Partially supported by Israel Science Foundation Grant 495/23.

Acknowledgements We thank Nabil Mustafa and Sergio Cabello for useful discussions that motivated the study reported in this paper.

1 Introduction

A typical range-searching problem asks to preprocess a set P of m points in $\mathbb{R}^d$ into a data structure so that for a query region σ, some aggregate statistics on $\sigma \cap P$ can be computed quickly, e.g., testing whether $\sigma \cap P = \emptyset$, computing $|\sigma \cap P|$, or computing a weighted sum of $\sigma \cap P$ (given a weight function on P). A central problem in computational geometry, range searching has been extensively studied over the last five decades, and sharp bounds are known for many instances; see [1, 2, 3, 10, 47] and references therein. For instance, a simplex range query (where the query region is a simplex) can be answered in $O^*(m/\omega^{1/d})$ time using $O^*(\omega)$

space and preprocessing for any $\omega \in [m, m^d]$, and (almost) matching lower bounds are known.[1] In particular, the total time spent, including the preprocessing cost, in answering a set Σ of n simplex range queries is $O^*((mn)^{1-\frac{1}{d+1}} + m + n)$. The known lower bounds imply that this bound is tight within a $\log^{O(1)} m$ factor. However, such sharp lower bounds are not known for more general classes of range queries. For instance, the best known data structures answer a disk range query (for points in the plane and disks as queries) in $O(m^{3/4}/\omega^{1/4} + \log m)$ time using $O(\omega)$ space and $O(\omega \log m)$ preprocessing, for any $\omega \in [m, m^3]$, and thus the total cost of answering n disk range queries is $O((m^{3/5}n^{4/5} + m + n) \log n)$, while the best known lower bound is $\Omega(m^{2/3}n^{2/3})$. (Slightly better lower bounds are known for annulus range queries [1].) A similar gap holds (see below for the exact bounds) for the more general class of semi-algebraic range queries.[2] A natural and fundamental open question is whether this gap can be narrowed. There is some evidence that the current upper bounds are not optimal.

Given a set P of m points and a set Γ of n surfaces in $\mathbb{R}^d$, the *incidence problem* on P and Γ asks for obtaining a sharp bound on the maximum number of *incidences*, i.e., pairs $(p, \gamma) \in P \times \Gamma$ such that $p \in \gamma$. Originally posed for bounding the number of incidences between points and lines in the plane [56], by now there is vast literature on this topic; see [14, 51, 54, 53, 55, 56] for a sample of references. There is a deep connection between range searching and the incidence problem. For example, many of the techniques developed for bounding incidences (e.g., geometric cuttings and polynomial partitioning techniques) have led to fast data structures for range searching, and vice versa. Similarly, many of the lower-bound constructions for range searching exploit the incidence structure between points and curves/surfaces [1]. As such, there is a general belief that the two problems are closely related, and that the running time of (at least off-line) range queries should be almost the same as the number of incidences between points and the corresponding curves/surfaces that bound these regions. This certainly holds for simplex range searching and for incidences between points and lines in $\mathbb{R}^2$, and, with some constraints, for points and halfspaces (for range searching) and hyperplanes (for incidences) in higher dimensions; see, e.g., [18, 27]. This also used to be the case for disk range searching and point-circle incidence problem – the best known upper bound on incidences between m points and n circles used to be $O(m^{3/5}n^{4/5} + m + n)$ (see, e.g., Pach and Sharir [50]). However, Aronov and Sharir [20], and later Agarwal et al. [14], have obtained an improved bound of $O^*(m^{2/3}n^{2/3} + m^{6/11}n^{9/11} + m + n)$ for point-circle incidences (see also [8, 53] for related results), and later Agarwal and Sharir [16] presented an algorithm for computing these incidences in the same time bound (up to the factor in the $O^*(\cdot)$ notation). More recently, Sharir and Zahl [53] obtained a bound of $O^*(m^{\frac{2s}{5s-4}}n^{\frac{5s-6}{5s-4}} + m^{2/3}n^{2/3} + m + n)$ on the number of incidences between m points and n semi-algebraic curves of constant complexity, where s is the number of degrees of freedom of the curves (the number of real parameters needed to specify a curve). If we believe the above conjecture, as we tend to, a natural question is whether one can obtain algorithms for disk range searching, and more broadly for semi-algebraic range searching, that have these running times, up to possible $O^*(\cdot)$ factors, at least in the off-line setting.

In this paper we answer this question in the affirmative for $d = 2$, by presenting an algorithm for the off-line semi-algebraic range-searching problem in $\mathbb{R}^2$, with (randomized expected) running time that almost matches (again, up to $O^*(\cdot)$ factors) the aforementioned

[1] Throughout this paper, the $O^*(\cdot)$ notation hides subpolynomial factors, typically of the form m^ε, and its associated ε-dependent constant of proportionality, for any $\varepsilon > 0$.

[2] Roughly speaking, a semi-algebraic set in $\mathbb{R}^d$ is the set of points in $\mathbb{R}^d$ satisfying a Boolean predicate over a set of polynomial inequalities; the complexity of the predicate and of the set is defined in terms of the number of polynomials involved and their maximum degree; see [23] for details.

incidence bounds. Our algorithm also works for off-line point-enclosure queries (see below) amid semi-algebraic sets in $\mathbb{R}^2$ within the same time bound. A recent result of Chan et al. [33] shows that two-dimensional m point-enclosure queries amid n semi-algebraic sets in an on-line context can be performed within the same bound as in the off-line setting discussed in this paper.

Problem statement. Let P be a set of m points in $\mathbb{R}^2$, let Σ be a set of n semi-algebraic sets of constant complexity in $\mathbb{R}^2$. Let s denote the *parametric dimension* of the regions in Σ, for some constant $s > 0$, meaning that each region can be specified by at most s real parameters. Let $(S, +)$ be a semigroup, and let $w : P \to S$ be a weight function. For a subset $R \subseteq P$, let $w(R) = \sum_{p \in R} w(p)$. Our goal is to compute $w(P \cap \sigma)$, for every $\sigma \in \Sigma$. This semigroup model encapsulates many popular variants of range searching [3]. Alternatively, we may assign a weight function $\eta : \Sigma \to S$ and compute, for every point $p \in P$, the weight $\sum_{\sigma \in \Sigma : p \in \sigma} \eta(\sigma)$. This dual setup is referred to as *point enclosure* searching.

To solve the above problems, and some of their variants, we formulate a more general problem: Let $\Phi : \Sigma \times P \to \{0, 1\}$ be the Boolean predicate such that, for $\sigma \in \Sigma$ and $P \in P$, $\Phi(\sigma, p) = 1$ iff $p \in \sigma$. Let $\Sigma \,\Phi\, P = \{(\sigma, p) \in \Sigma \times P \mid \Phi(\sigma, p) = 1\}$. A popular method of representing $\Sigma \,\Phi\, P$ compactly is to use a *biclique partition* $\mathcal{B}_\Phi := \mathcal{B}_\Phi(\Sigma, P) = \{(\Sigma_1, P_1), \ldots, (\Sigma_u, P_u)\}$, where $\Phi(\sigma, p) = 1$ for all pairs $(\sigma, p) \in \Sigma_i \times P_i$, and for any pair $(\sigma, p) \in \Sigma \times P$ with $\Phi(\sigma, p) = 1$, there is a unique $i \leq u$ such that $(\sigma, p) \in \Sigma_i \times P_i$. The *size* of $\mathcal{B}_\Phi$, denoted by $|\mathcal{B}_\Phi|$, is defined to be $\sum_{i=1}^{u} (|\Sigma_i| + |P_i|)$. Given $\mathcal{B}_\Phi$, both off-line range-searching and point-enclosure problems can be solved in $O(|\mathcal{B}_\Phi|)$ time. We thus focus on computing $\mathcal{B}_\Phi$, which is useful for other problems as well – see below.

Related work. We refer the reader to the survey papers [3, 10, 47] for a review of range-searching. The best-known data structures for semi-algebraic range searching can answer a query, on an input set of m points, in $O^*(m^{1-1/d})$ time using $O(m)$ space, or in $O(\log m)$ time using $O^*(m^s)$ space, where s is the parametric dimension of the query ranges [13, 49, 6]. By combining these data structures, in a so-called space/query-time tradeoff, for $\omega \in [m, m^s]$, a semi-algebraic range query can be answered in $O^*((m/\omega^{1/s})^{\frac{1-1/d}{1-1/s}})$ time using $O^*(\omega)$ space and preprocessing, and thus the total time taken (including preprocessing cost) in answering n semi-algebraic queries is $O^*(n^{\frac{1-1/s}{1-1/ds}} m^{\frac{1-1/d}{1-1/ds}} + m + n)$ [5]. Afshani and Chang [1, 2] showed that any data structure of size ω needs $\Omega^*((n^s/\omega)^{1/\rho})$ time, where $\rho = (s^2+1)(s-1)$, to answer a 2D semi-algebraic range-reporting query in the pointer machine model. They also showed that if P is a set of random points in $\mathbb{R}^d$, a range query can be answered in $O((n^s/\omega)^{\frac{1}{3s-4}})$ time.

The problem of representing a graph compactly using cliques or bicliques has been studied for at least four decades [36, 57]. For an arbitrary graph with n vertices, the worst-case bound on the size of the smallest biclique partition (again, the size of the partition is the sum of the sizes of the vertex sets of its bicliques) is $\Theta(n^2/\log n)$ [57]. However, significantly better bounds are known for geometric graphs, where the vertices are geometric objects (such as points, disks, segments, etc.) and two vertices are connected by an edge if the corresponding objects satisfy some geometric relation (such as two objects intersect, or be within distance r, for some parameter r). For example, interval graphs on n intervals on the line admit a biclique partition of size $O(n \log n)$, point-orthogonal-rectangle-incidence graphs in $\mathbb{R}^d$ admit such a representation of size $O((m+n)\log^{O(1)} n)$, unit-disk and segment-intersection graphs have a representation of size $O^*(n^{4/3})$ [17, 46], and point-hyperplane

incidence graphs admit an $O^*((mn)^{1-1/d} + m + n)$ representation size [18]. Recently, there has been some work on bounding the size of biclique partitions of general semi-algebraic geometric graphs (whose vertices are points in $\mathbb{R}^d$ and whose edges are defined by a semi-algebraic predicate) [5, 39]. We note though that not all geometric graphs, even in the plane, admit a small-size bipartite clique cover [4]. Biclique partitions/covers have been effectively applied to study extremal properties of geometric graphs, such as the regularity lemma, Zarankiewicz's problem, etc. [39, 41, 42, 43]. Most algorithms for computing these biclique partitions are based on off-line range-searching techniques; see, e.g., [15, 17, 46], affirming the close relationship between incidence and range-searching problems.

In addition, faster algorithms for some basic graph problems have been proposed using biclique partitions (their running time being faster than what one could have obtained by running them on an explicit representation of the graph) [12, 17, 29, 40]. For example, BFS/DFS can be implemented in $O(N)$ time [12, 17] and a maximum bipartite matching in an intersection graph can be computed in $O^*(N)$ time [29], assuming that a biclique partition of size N is given. The applicability of biclique partitions, however, goes far beyond basic graph algorithms. For example, the multipole algorithms for the so-called n-body problem, developed in the 1980's, can be regarded as an application of biclique partition of the complete graph of a set of points, where each biclique is *well-separated*. Building on, and extending, this idea, Callahan and Kosaraju [31, 30] introduced the notion of *well-separated pair decomposition* (WSPD), showed the existence of small-size WSPD for point sets in $\mathbb{R}^d$, and applied such decompositions to develop faster algorithms for many geometric proximity problems. Biclique partitions of geometric graphs also have been extensively used for a range of geometric optimization problems [7, 15, 17, 46, 48].

Our results. The main result of this paper is stated in the following theorem.

▶ **Theorem 1.** *Let P be a set of m points in $\mathbb{R}^2$, and let Σ be a set of n semi-algebraic regions in $\mathbb{R}^2$ with parametric dimension s for some constant $s > 0$. Let $\Phi : \Sigma \times P \to \{0, 1\}$ be the Boolean predicate such that $\Phi(\sigma, p) = 1$ iff $p \in \sigma$. A biclique partition of $\Sigma\,\Phi\,P$ of size $O^*\left(m^{\frac{2s}{5s-4}} n^{\frac{5s-6}{5s-4}} + m^{2/3}n^{2/3} + m + n\right)$ can be computed within the same randomized expected time (up to a subpolynomial factor).*

This immediately implies the following corollary:

▶ **Corollary 2.** *Let P be a set of m points in $\mathbb{R}^2$, let Σ be a set of n semi-algebraic regions in $\mathbb{R}^2$ with parametric dimension s for some constant $s > 0$, let $(S, +)$ be semigroup, and let $w : P \to S$ be a weight function. The weight $w(\sigma \cap P)$, for every $\sigma \in \Sigma$, can be computed in $O^*\left(m^{\frac{2s}{5s-4}} n^{\frac{5s-6}{5s-4}} + m^{2/3}n^{2/3} + m + n\right)$ randomized expected time. Conversely, given a weight function $\eta : \Sigma \to S$, the weight $\sum_{\sigma\in\Sigma:\sigma\ni p} \eta(\sigma)$, for every $p \in P$, can be computed within the same time bound.*

Our main observation is that the boundary arcs of the regions in Σ can be processed to yield a family Ψ of $O^*(n^{3/2})$ pseudo-trapezoids, each bounded by (up to) two vertical lines and two subarcs of boundaries of regions in Σ, such that the edges of Ψ are pseudo-segments, i.e., any pair of edges of Ψ intersect in at most one point. Using the duality transform for pseudo-lines, proposed by Agarwal and Sharir [16], we first present (in Section 2) an algorithm for computing a biclique partition of $\Psi\Phi P$ of size $O^*(m\sqrt{n} + n^{3/2})$. Using a standard hierarchical-cutting based method [16], we improve (in Section 3) the size of the biclique partition to $O^*(m^{2/3}n^{2/3} + n^{3/2})$, or even further to $O^*(m^{2/3}\chi^{1/3} + n^{3/2})$, where χ

is the number of intersections between the curves. Finally, by working in the s-dimensional parametric space of Σ, we further improve the bound on the size of the biclique partition to $O^*(m^{\frac{2s}{5s-4}}n^{\frac{5s-6}{5s-4}} + m^{2/3}n^{2/3} + m + n)$ (Section 4).

We conclude the discussion on our contributions by mentioning two further applications of our results. The first is efficiently obtaining approximate solutions for the geometric *hitting set* and *set cover* problems, and the latter concerns faster implementations of basic graph algorithms for geometric proximity graphs. Due to lack of space we defer these details to the full version [11].

2 Bicliques Using Pseudo-Line Duality: The First Step

Let Ψ be a set of n pseudo-trapezoids in $\mathbb{R}^2$, each bounded from above and below by x-monotone semi-algebraic arcs with parametric dimension $s > 0$, for some constant $s > 0$, and from left and right by two vertical lines (some of these boundary arcs and lines may be absent). Furthermore, we assume that each pair of these arcs intersect in at most one point, i.e., the upper and lower edges of the pseudo-trapezoids in Ψ form a collection of *pseudo-segments*. Let P be a set of m points in $\mathbb{R}^2$. Let $\Psi\,\Phi\,P \subseteq \Psi \times P$ be the set of pairs (ψ, p) such that $p \in \psi$. The main result of this section is a randomized algorithm, with $O^*(m\sqrt{n}+n)$ expected running time, that constructs a biclique partition $\mathcal{B} := \mathcal{B}_\Phi(\Psi, P)$ of $\Psi\Phi P$ of size $O((m\sqrt{n}+n)\log^3 n)$. We first give an overview of the algorithm, then describe its main steps in detail, and finally analyze its performance. This algorithm serves as the innermost routine in our overall algorithm.

2.1 Overview of the algorithm

We begin by defining two Boolean predicates $\Phi^\uparrow, \Phi^\downarrow : \Psi \times P \to \{0, 1\}$ such that $\Phi^\uparrow(\psi, p) = 1$ (resp., $\Phi^\downarrow(\psi, p) = 1$) if p lies vertically above (resp., below) the bottom (resp., top) arc of ψ. Note that $\Phi(\psi, p) = \Phi^\uparrow(\psi, p) \wedge \Phi^\downarrow(\psi, p)$.

The algorithm consists of the following high-level steps:

- **(i)** We construct a segment tree T on the x-projections of the pseudo-trapezoids in Ψ. Each node v of T is associated with a vertical slab $W_v = I_v \times \mathbb{R}$. A pseudo-trapezoid $\psi \in \Psi$ is stored at v if the x-projection of ψ contains I_v but does not contain $I_{p(v)}$, where $p(v)$ is the parent of v. Let $\Psi_v \subseteq \Psi$ be the set of pseudo-trapezoids stored at v, clipped to within W_v, and let $P_v = P \cap W_v$. Set $n_v = |\Psi_v|$ and $m_v = |P_v|$.
- **(ii)** For each node v of T, we compute a biclique partition $\mathcal{B}_v := \mathcal{B}_\Phi(\Psi_v, P_v)$ of $\Psi_v\Phi P_v$, as follows. We partition P_v into $r_v = \lceil m_v/\sqrt{n_v}\rceil$ subsets $P_v^{(1)}, \ldots, P_v^{(r_v)}$ of size at most $\sqrt{n_v}$ each. Set $m_{v,i} = |P_v^{(i)}| \le \sqrt{n_v}$. We compute a biclique partition $\mathcal{B}_{v,i} := \mathcal{B}(\Psi_v, P_v^i)$ for every $i \le r_v$, in (the following) two stages.
  - **(ii.a)** For every node $v \in T$ and for every $i \le r_v$, we compute a biclique partition $\mathcal{B}^\uparrow_{v,i} := \mathcal{B}_{\Phi^\uparrow}(\Psi_v, P_v^i)$.
  - **(ii.b)** Next, for each biclique $(\Psi_j, P_j) \in \mathcal{B}^\uparrow_{v,i}$, we compute a biclique partition $\mathcal{B}_{v,i,j} := \mathcal{B}_{\Phi^\downarrow}(\Psi_j, P_j)$ of $\Psi_j\Phi^\downarrow P_j$. We set $\mathcal{B}_{v,i} = \bigcup_{(\Psi_j,P_j)\in\mathcal{B}^\uparrow_{v,i}} \mathcal{B}_{v,i,j}$.
- **(iii)** We set $\mathcal{B}_v = \bigcup_{i=1}^{r_v} \mathcal{B}_{v,i}$ and return $\mathcal{B} = \bigcup_{v\in T} \mathcal{B}_v$ as the desired biclique partition $\mathcal{B}_\Phi(\Psi, P)$.

Steps (ii.a) and (ii.b) are the only nontrivial steps in the above algorithm. We describe the algorithm for Step (ii.a). A symmetric procedure can be used for Step (ii.b).

2.2 Biclique partition for $\Phi^{\uparrow}$

Let W be a vertical slab. Let Γ be a set of n x-monotone semi-algebraic arcs with parametric dimension s, for some constant $s > 0$, whose endpoints lie on the boundary lines of W, and any pair of arcs in Γ intersect in at most one point, i.e., Γ is a set of pseudo-segments. Let $P \subset W$ be a set of m points. Abusing the notation, let $\Phi^{\uparrow} : \Gamma \times P \to \{0, 1\}$ be a Boolean predicate such that $\Phi^{\uparrow}(\gamma, p) = 1$ if p lies above γ and 0 otherwise. We describe a randomized algorithm, with expected running time $O(m^2 + n \log n)$, for computing a biclique partition $\mathcal{B}$ of $\Gamma\Phi^{\uparrow}P$ of size $O(m^2 + n \log n)$. By choosing P to be P_v^i and Γ to be the set of bottom arcs of trapezoids in Ψ_v, we compute $\mathcal{B}_{\Phi^{\uparrow}}(\Psi_v, P_v^i)$, as required in Step (ii.a).

Our algorithm is built on two previous algorithms. First, we rely on the pseudo-line duality transform described by Agarwal and Sharir [16], as a major tool for the construction of the desired biclique partition (see also [44]). The duality transform maps the arcs in Γ to a set Γ^* of dual points lying on the x-axis, and the points in P to a set P^* of dual x-monotone curves, such that p lies above (resp., on, below) γ if and only if the dual curve p^* passes above (resp., through, below) the dual point γ^*. Furthermore, P^* is a set of pseudo-lines, i.e., each pair of them intersect at most once. Agarwal and Sharir describe an $O^*(m^2 + n)$-time sweep-line algorithm to construct P^* and to compute a DCEL representation [24] of the arrangement $\mathcal{A}(P^*)$, as well as the subset $\Gamma_f^* \subset \Gamma^*$ of dual points lying in each face f of $\mathcal{A}(P^*)$. Let $\gamma_1, \ldots, \gamma_n$ be the ordering of arcs in Γ in increasing order of the y-coordinates of their left endpoints, then the x-coordinate of the dual point γ_i^* is i. Conversely, the dual curves are ordered in the $(+y)$-direction at $x = -\infty$ in the decreasing order of the x-coordinates of the primal points; see [16]. We note that the curves in P^* do not have constant combinatorial (or geometric) complexity, as each of them may contain many breakpoints and turns, in which it weaves its way above and below the dual points of Γ^* on the x-axis. Nevertheless, we never need an explicit representation of a dual curve. The representation computed by the algorithm in [16] enables us to compute (i) the vertical ordering of a pair of curves at any given x-coordinate, and (ii) the (unique) intersection point between any pair of curves, in $O(1)$ time.

Second, we use the randomized algorithm by Clarkson and Shor [38] for constructing the vertical decomposition $\mathcal{A}^{\shortparallel}(P^*)$ of $\mathcal{A}(P^*)$, from which we will obtain the desired biclique partition. Although one can combine the ideas of the two algorithms and work directly in the primal plane, we describe, for the simplicity of exposition and analysis, the construction in two stages – namely, we first compute $\mathcal{A}(P^*)$ using the sweep-line algorithm of [16], so that we have P^* (implicitly) at our disposal, and then we run the algorithm of Clarkson and Shor on P^*. Since we will be adapting the Clarkson-Shor algorithm to compute the biclique partition, we describe the algorithm briefly, but first a few notations.

For a subset $R^* \subseteq P^*$, let $\mathcal{A}^{\shortparallel}(R^*)$ denote the vertical decomposition of the arrangement $\mathcal{A}(R^*)$, i.e., we draw rays in both $(+y)$- and $(-y)$-directions from every vertex of $\mathcal{A}(R^*)$ until it intersects another curve, or all the way to $\pm\infty$. $\mathcal{A}^{\shortparallel}(R^*)$ partitions each face of $\mathcal{A}(R^*)$ into pseudo-trapezoids, which we refer to as *cells*, to distinguish them from the input pseudo-trapezoids. Let $\mathcal{F} = \mathcal{F}_{R^*}$ be the family of all cells that appear in $\mathcal{A}^{\shortparallel}(R^*)$, for some specific subset $R^* \subset P^*$. By construction, each cell $\tau \in \mathcal{F}$ is defined by a set $\mathcal{D}(\tau)$ of at most four curves of P^*. Let $\mathcal{K}(\tau) \subseteq P^*$ denote the subset of curves that intersect the interior of τ, the so-called *conflict list* of τ. It is well known (see, e.g., [38]) that τ appears in $\mathcal{A}^{\shortparallel}(R^*)$ if and only if $\mathcal{D}(\tau) \subset R^*$ and $\mathcal{K}(\tau) \cap R^* = \emptyset$.

Let $p_1^*, \ldots, p_m^*$ be a random permutation of P^*, and let $P_i^* = \{p_1^*, \ldots, p_i^*\}$. The Clarkson-Shor algorithm maintains the invariant that after processing the first i curves, it has computed (i) $\mathcal{A}^{\shortparallel}(P_i^*)$, represented by its dual graph, (ii) the conflict list $\mathcal{K}(\tau)$ of each cell in $\tau \in \mathcal{A}^{\shortparallel}(P_i^*)$,

and (iii) for every dual curve $p_j^* \in P^* \setminus P_i^*$, the set $\mathcal{L}(p_j^*)$ of cells of $\mathcal{A}^{\shortparallel}(P_i^*)$ that p_j^* crosses, i.e., $\mathcal{L}(p_j^*) = \{\tau \in \mathcal{A}^{\shortparallel}(P_i^*) \mid p_j^* \in \mathcal{K}(\tau)\}$, sorted along p_j^*. Suppose we have inserted $p_1^*, \ldots, p_{i-1}^*$ and computed $\mathcal{A}(P_{i-1}^*)$ and $\mathcal{A}^{\shortparallel}(P_{i-1}^*)$, plus the auxiliary structures just mentioned. In the i-th step, we process the cells of $\mathcal{L}(p_i^*)$ in order along p_i^*. For each $\tau \in \mathcal{L}(p_i^*)$, we partition τ into $O(1)$ pseudo-trapezoids (new cells), by the pattern in which p_i^* crosses τ, and compute the conflict list of each new cell from $\mathcal{K}(\tau)$. This step requires (i) comparing the vertical ordering of curves in $\mathcal{K}(\tau)$ with p_i^* at the left or right edge of τ, and (ii) testing whether a curve of $\mathcal{K}(\tau)$ intersects p_i^* in a given x-interval. As mentioned above, both of these steps can be done in $O(1)$ time using the information computed by the sweep-line algorithm. After this step, we have a refinement of $\mathcal{A}^{\shortparallel}(P_i^*)$ – each cell (obtained by the split induced by p_i^*) lies in a cell of $\mathcal{A}^{\shortparallel}(P_i^*)$, but multiple cells of the refinement might lie in the same cell of $\mathcal{A}^{\shortparallel}(P_i^*)$. To address this issue, the algorithm performs a clean-up step in which it merges the newly created cells if they lie in the same cell of $\mathcal{A}^{\shortparallel}(P_i^*)$ (when we merge two cells, we also merge their conflict lists). When the iteration over i terminates, we obtain $\mathcal{A}^{\shortparallel}(R^*)$. The total time spent in inserting p_i^* is $O\left(\sum_{\tau \in \mathcal{L}(p_i^*)} |\mathcal{K}(\tau)|\right)$. The fairly straightforward details of the algorithm, and the accompanying analysis of its expected running time, can be found in [25, 38, 52].

Let Ξ be the set of cells created by the algorithm over all steps, i.e., $\Xi = \bigcup_{i=1}^{n} \mathcal{A}^{\shortparallel}(P_i^*)$. We also add $\tau_0 = \mathbb{R}^2$ as a cell to Ξ, which is the initial cell before any of the curves were inserted. For a pair of cells $\tau, \tau' \in \Xi$, we say τ' is a *child cell* of τ if $\tau \cap \tau' \neq \emptyset$ and there exists an $i < m$ such that $\tau \in \mathcal{A}^{\shortparallel}(P_{i-1}^*)$, $\tau' \in \mathcal{A}^{\shortparallel}(P_i^*) \setminus \mathcal{A}^{\shortparallel}(P_{i-1}^*)$; τ is called the *parent* of τ'. Note that a cell τ' may have many parents, but a cell has only $O(1)$ children. We construct a *history DAG* $\mathcal{G} = (\Xi, \mathcal{E})$, where $\tau' \to \tau \in \mathcal{E}$ if τ' is a parent of τ.

For each cell $\tau \in \Xi$, we construct a *canonical subset* $\mathcal{C}(\tau) \subseteq P^*$, the set of curves that appear in the conflict lists of its parents and that lie *above* τ (without intersecting it), i.e.,

$$\mathcal{C}(\tau) = \{p_i^* \mid p_i^* \text{ lies above } \tau,\ p_i^* \in \mathcal{K}(\hat{\tau}),\ \text{ and } \hat{\tau} \to \tau \in \mathcal{E}\}.$$

Set $P_\tau = \{p_i \mid p_i^* \in \mathcal{C}(\tau)\}$. Since the out-degree of each node in $\mathcal{G}$ is $O(1)$, for any $r \geq 1$,

$$\sum_{\tau \in \Xi} |\mathcal{C}(\tau)|^r = O\left(\sum_{\tau \in \Xi} |\mathcal{K}(\tau)|^r\right). \tag{1}$$

Using the information computed by the Agarwal-Sharir algorithm [16], we can check in $O(1)$ time, for each curve $p_i^* \in \mathcal{K}(\hat{\tau})$, whether p_i^* lies above τ. Hence, the total time spent in computing P_τ, over all $\tau \in \Xi$, is $O\left(\sum_{\tau \in \Xi} |\mathcal{K}(\tau)|\right)$.

Next, for a cell $\tau \in \Xi$, we set $\Gamma_\tau = \{\gamma \in \Gamma \mid \gamma^* \in \tau\}$. We compute Γ_τ as follows. For every (dual) point $\gamma^* \in \Gamma^*$, we traverse a path Π_γ in $\mathcal{G}$, starting from the root until we reach the leaf (the cell of $\mathcal{A}^{\shortparallel}(P^*)$) that contains γ^*, such that $\gamma^* \in \tau$ for every cell τ on Π_γ, as follows. Suppose we are at a node τ that contains γ^*. We check which of its $O(1)$ children contains γ^*. This step requires testing whether γ^* lies inside a child cell τ'. We can easily determine in $O(1)$ time whether γ^* lies to the left (resp., to the right) of the left (resp., right) vertical edge of τ', but the top/bottom edge of τ' may have large complexity (due to the "erratic" way in which the dual arrangement is constructed in [16]). However, the top (or bottom) arc is a portion of a dual curve p_i^*, and the duality transform ensures that γ^* lies below/above p_i^* if and only if γ lies below/above p_i. Since γ is a semi-algebraic arc of constant complexity, we can test the above/below relationship between p_i and γ in $O(1)$ time. Hence, Π_γ can be computed in $O(|\Pi_\gamma|)$ time. For each cell $\tau \in \Pi_\gamma$, we add γ to τ. We return $\mathcal{B}_{\Phi^\uparrow} := \{(\Gamma_\tau, P_\tau) \mid \tau \in \Xi\}$ as the desired biclique partition of $\Gamma\Phi^\uparrow P$.

▶ **Lemma 3.** *$\mathcal{B}_{\Phi\uparrow}$ is a biclique partition of $\Gamma\Phi^{\uparrow}P$.*

Proof. By construction and the property of the dual transform, it is clear that all points of P_τ lie above all the arcs in Γ_τ. Conversely, let $(\gamma, p) \in \Gamma \times P$ be a pair such that p lies above γ. By construction, $p^* \in \mathcal{K}(\tau_0)$ since $\tau_0 = \mathbb{R}^2$ and $\mathcal{K}(\tau) = \emptyset$ for the leaf $\tau \in \Pi_\gamma$. Let $\hat{\tau}$ be the lowest node in the path Π_γ for which $p^* \in \mathcal{K}(\hat{\tau})$; $\hat{\tau}$ is a non-leaf node. Let $\sigma \in \Pi_\gamma$ be the child of $\hat{\tau}$ in Π_γ. Since $p^* \notin \mathcal{K}(\sigma)$, $\gamma^* \in \sigma$, and p^* lies above γ^*, we conclude that p^* lies above σ and $p^* \in \mathcal{C}(\sigma)$. Hence, $(\gamma, p) \in \Gamma_\sigma \times P_\sigma$. Furthermore, σ is the only cell in Π_γ for which $p^* \in \mathcal{C}(\sigma)$, therefore there is a unique biclique in $\mathcal{B}_{\Phi\uparrow}$ that contains the pair (γ, p), implying that it is a biclqiue partition of $\Gamma\Phi^{\uparrow}P$. ◀

We now bound the size of $\mathcal{B}_{\Phi\uparrow}$ and the expected running time of the algorithm. It follows from the above discussion that the total size of $\mathcal{B}_{\Phi\uparrow}$ is

$$|\mathcal{B}_{\Phi\uparrow}| = O\Bigg(\sum_{\tau\in\Xi} |\mathcal{K}(\tau)| + \sum_{\gamma\in\Gamma} |\Pi_\gamma|\Bigg). \tag{2}$$

Since the randomized incremental construction adheres to the Clarkson-Shor framework, we can follow the standard analysis of a randomized incremental construction in the Clarkson-Shor framework, as given in [25, 38, 52], from which the following properties hold:

(P1) The expected depth of the history DAG $\mathcal{G}$ is $O(\log m)$.

(P2) For any $t \le m$, let $\Xi_t = \mathcal{A}^{\shortparallel}(P_t^*) \setminus \mathcal{A}^{\shortparallel}(P_{t-1}^*)$ be the set of cells that were created in the i-th step of the algorithm. For any $1 \le t \le m$ and for any $d \ge 1$, we have

$$\mathsf{E}\Bigg[\sum_{\tau\in\Xi_t} |\mathcal{K}(\tau)|^d\Bigg] = O\left(\frac{1}{t}\left(\frac{m}{t}\right)^d t^2\right) = O\left(\frac{m^d}{t^{d-1}}\right). \tag{3}$$

See, e.g., [38, 25]. We will be using (3) for $d = 1, 2$, for which the expected values are $O(m)$ and $O(m^2/t)$, respectively.

Plugging the bounds in (P1) and (P2) into (2), we obtain:

$$\mathsf{E}[|\mathcal{B}_{\Phi\uparrow}|] = O\Bigg(\sum_{t=1}^{m} \mathsf{E}\Bigg[\sum_{\tau\in\Xi_t} |\mathcal{K}(\tau)|\Bigg] + n\log m\Bigg) = O\Bigg(\sum_{t=1}^{m} m + n\log m\Bigg) = O(m^2 + n\log m).$$

If the size of $\mathcal{B}(\Gamma, P) \ge c(m^2 + n\log m)$, for some sufficiently large constant $c > 0$, we reconstruct $\mathcal{B}_{\Phi\uparrow}$. A similar argument then shows that the expected running time of the algorithm is $O(m^2 + n\log m)$. We thus obtain the following lemma:

▶ **Lemma 4.** *Let Γ be a set of n x-monotone semi-algebraic arcs in $\mathbb{R}^2$ of constant complexity, whose endpoints lie on the boundary lines of a vertical slab W, and any pair of arcs in Γ intersect in at most one point, i.e., Γ is a set of pseudo-segments. Let $P \subset W$ be a set of m points. Then a biclique partition of $\Gamma\,\Phi^{\uparrow}\,P$ of size $O(m^2 + n\log m)$ can be computed in expected time $O(m^2 + n\log m)$.*

2.3 Putting it all together

Returning to the problem of computing a biclique partition of $\Psi\,\Phi\,P$, let v be a node of the segment tree T, and let Ψ_v and $P_v^{(1)}, \ldots, P_v^{(r_v)}$ be the sets as defined above. Set $n_v = |\Psi_v|$ and $m_v = |P_v|$. For a pseudo-trapezoid $\psi_a \in \Psi_v$, let γ_a^-, γ_a^+ be its bottom and top boundary arcs, respectively. By construction, the endpoints of γ_a^-, γ_a^+ lie on the boundary lines of the vertical slab W_v, so ψ_a also straddles W_v. Let $\Gamma_v^- = \{\gamma_i^- \mid \psi_i \in \Psi_v\}$ be the set of bottom

arcs of the pseudo-trapezoids in Ψ_v. Fix a value $1 \le i \le r_v$. We first compute a biclique partition $\mathcal{B}^{\uparrow}_{v,i}$ of $\Gamma^-_v \,\Phi^{\uparrow}\, P^{(i)}_v$ using the above algorithm. Let (Γ^-_j, P_j) be a biclique in this partition, and let Γ^+_j be the set of top arcs of the pseudo-trapezoids whose bottom arcs are in Γ^-_j, i.e., $\Gamma^+_j = \{\gamma^+_a \mid \gamma^-_a \in \Gamma^-_j\}$. Following the above algorithm (but reversing the direction of the y-axis), we compute a biclique partition $\mathcal{B}_{v,i,j}$ of $\Gamma^+_j \,\Phi^{\downarrow}\, P_j$. For each resulting biclique $(\Gamma^+_{j,t}, P_{j,t})$, we replace $\Gamma^+_{j,t}$ with $\Psi_{j,t} \subseteq \Psi$, the set of trapezoids whose top arcs are in Γ^+_j. Abusing the notation a little, let $\mathcal{B}_{v,i,j}$ denote the resulting biclique partition. We repeat this step for all bicliques in $\mathcal{B}(\Gamma^-_v, P^{(i)}_v)$, and set $\mathcal{B}_{v,i} = \bigcup_{(\Gamma^-_j, P_j)\in\mathcal{B}^{\uparrow}_{v,i}} \mathcal{B}_{v,i,j}$, and return $\mathcal{B}_{v,i}$ as a biclique partition of $\Psi_v \,\Phi\, P^{(i)}_v$. By repeating this step for all $i \le r_v$ and for all $v \in T$, we obtain the desired biclique partition $\mathcal{B} := \mathcal{B}_{\Phi}(\Psi, P)$. It is easy to check that, by construction, the resulting collection of bicliques is edge disjoint, and its union gives all pairs (p, σ) with $p \in \sigma$, so it is indeed a desired biclique partition. It remains to bound its size.

By Lemma 4, $|\mathcal{B}_{v,i,j}| = O(|\Gamma^+_j| + |P_j|^2 \log |P_j|)$. Let $\mathcal{G}_{v,i} = (\Xi_{v,i}, \mathcal{E}_{v,i})$, where $\Xi_{v,i}$ is the set of cells created by the algorithm while constructing $\mathcal{A}^{\shortparallel}(P^*_{v,i})$, be the history DAG constructed by the algorithm (invoked on Γ_v and $P^{(i)}_v$) for computing $\mathcal{B}_{v,i}$. Recall that each biclique (Γ_j, P_j) in $\mathcal{B}_{v,i}$ corresponds to a cell $\tau \in \Xi_{v,i}$. Hence, for each cell (node) $\tau \in \mathcal{G}_{v,i}$, we compute a biclique partition of size $O(m^2_\tau + n_\tau \log m_\tau)$, where $m_\tau = |P_\tau| = |\mathcal{C}(\tau)|$ and $n_\tau = |\Gamma_\tau|$. Summing over all cells of Ξ and using (1), the total size of the biclique partition of $\mathcal{B}_{\Phi}(\Gamma_v, P^{(i)}_v)$ is

$$\sum_{\tau\in\Xi} O(m^2_\tau + n_\tau \log m) = \sum_{\tau\in\Xi} O\big(|\mathcal{K}(\tau)|^2 + n_\tau \log m\big).$$

The same argument as above implies that

$$\mathsf{E}\left[\sum_{\tau\in\Xi} n_\tau\right] = O(n_v \log m_{v,i})$$

(recall that $m_{v,i} = |P^{(i)}_v|$). By (3),

$$\mathsf{E}\left[\sum_{\tau\in\Xi} |\mathcal{K}(\tau)|^2\right] = \mathsf{E}\left[\sum_{t=1}^{m} \sum_{\tau\in\Xi_t} |\mathcal{K}(\tau)|^2\right] = O\left(\sum_{t=1}^{m_{v,i}} \frac{m^2_{v,i}}{t}\right) = O(m^2_{v_i} \log m_{v,i}) = O(n_v \log n_v),$$

because $m_{v,i} \le \sqrt{n_v}$. Hence, the size of $\mathcal{B}_{v,i}$ is $O(n_v \log^2 n_v)$. Summing over all $i \le r_v = \lceil \frac{m_v}{\sqrt{n_v}} \rceil$, the size of $\mathcal{B}_v$ is $O((m_v\sqrt{n_v} + n_v)\log^2 n_v)$. Again, reconstructing $\mathcal{B}_v$ if its size exceeds $c_1((m\sqrt{n} + n)\log^2 n$, for an appropriate constant c_1, we obtain a biclique partition of size $O((m\sqrt{n} + n)\log^2 n)$.

▶ **Lemma 5.** *Let Ψ be a set of n pseudo-trapezoids in $\mathbb{R}^2$, each bounded from above and below by x-monotone semi-algebraic arcs of constant complexity, such that any pair of these arcs intersect in at most one point, and whose vertical edges lie on the boundary lines of a vertical slab W. Let P be a set of m points lying in W. Then a biclique partition of $\Psi \,\Phi\, P$ of size $O((m\sqrt{n} + n)\log^2 n)$ can be computed in expected time $O((m\sqrt{n} + n)\log^2 n)$.*

Finally, summing the size of biclique partitions over all nodes v of the segment tree and plugging the values $\sum_{v\in T} m_v = O(m \log n)$, $\sum_{v\in T} n_v = O(n \log n)$, we conclude:

▶ **Corollary 6.** *Let Ψ be a set of n pseudo-trapezoids in $\mathbb{R}^2$, each bounded from above and below by x-monotone semi-algebraic arcs of constant complexity, such that any pair of these arcs intersect in at most one point, and let P be a set of m points in $\mathbb{R}^2$. Then a biclique partition of $\Psi \,\Phi\, P$ of size $O((m\sqrt{n} + n)\log^3 n)$ can be computed in expected time $O((m\sqrt{n} + n)\log^3 n)$.*

3 Bicliques Using Cuttings: The Second Step

Let P be a set of m points in $\mathbb{R}^2$, and let Σ be a set of n semi-algebraic sets of constant complexity in $\mathbb{R}^2$, as defined in the introduction. Our goal is to compute a biclique partition $\mathcal{B}_\Phi(\Sigma, P)$ for the inclusion predicate Φ, i.e., $\Phi(\sigma, p) = 1$ if $p \in \sigma$. Let Γ denote the set of boundary edges of regions in Σ, each of which is a semi-algebraic arc of constant complexity.

Following the technique in [53] (see also [21]), we cut the arcs in Γ into $O^*(n^{3/2})$ subarcs that constitute a family of pseudo-segments, i.e., each pair of subarcs intersect at most once. Agarwal *et al.* [6] (see also [19]) presented an efficient algorithm for constructing these cuts that runs in $O^*(n^{3/2})$ time. Without loss of generality, we assume that each subarc is x-monotone. This step partitions the edges of each region $\sigma \in \Sigma$ into subarcs, which we view as new edges of σ. We compute the vertical decomposition of σ, which divides σ into a set of pseudo-trapezoids and further partitions its edges into smaller pieces. Each resulting pseudo-trapezoid is bounded by at most two vertical edges and two (top and bottom) semi-algebraic arcs that are portions of edges of σ. Let Ψ denote the resulting set of pseudo-trapezoids, and let Γ denote the set of their top and bottom edges. Set $|\Psi| = N$, so $|\Gamma| \leq 2N$; by construction, $N = O^*(n^{3/2})$. Let χ denote the number of intersection points between the arcs of Γ; $\chi = O(n^2)$. It suffices to construct a biclique partition for $\Psi\,\Phi\,P$ (that is, for Ψ instead of Σ). The algorithm described in the previous section already computes a biclique partition of size $O((m\sqrt{N} + N)\log^3 N) = O^*(mn^{3/4} + n^{3/2})$, within the same expected time. In this section, we show how to improve the bound to $O^*(m^{2/3}\chi^{1/3} + n^{3/2})$, using hierarchical cuttings [16, 34] in the primal plane. This step is analogous to the widely used approach for obtaining sharp bounds on various substructures of arrangements of curves in the plane or for the number of incidences between points and curves in the plane (see e.g. [16, 37, 52]).

Abstracting the setup for a moment, let X be a set of n x-monotone semi-algebraic arcs of constant complexity in $\mathbb{R}^2$, let χ be the number of vertices in $\mathcal{A}(\mathsf{X})$, and let Δ be a pseudo-trapezoid, within which we want to apply our construction. For a parameter $r > 1$, a partition of Δ into a family Ξ of pseudo-trapezoids, referred to as *cells*, to distinguish them, as before, from the input pseudo-trapezoids, is called a $(1/r)$-*cutting* of X within (or with respect to) Δ if every cell of Ξ is crossed by at most n/r arcs of X. As before, the *conflict list* of a cell $\tau \in \Xi$, denoted by X_τ, is the subset of arcs that cross τ. We follow Chazelle's hierarchical-cutting algorithm [34] to construct a $(1/r)$-cutting Ξ of X within Δ: we choose a sufficiently large constant r_0 and set $\nu = \lceil \log_{r_0} r \rceil$. We construct a sequence of cuttings $\Xi_0 = \Delta, \Xi_1, \ldots, \Xi_\nu = \Xi$ where Ξ_i is a $(1/r_0^i)$-cutting of X within Δ. Ξ_i is obtained from Ξ_{i-1} by computing for each $\tau \in \Xi_{i-1}$ a $(1/r_0)$-cutting of $\mathcal{A}(\mathsf{X}_\tau)$ within τ. By following the analysis in [34], it can be shown that the size of Ξ_i is bounded by $c_1((c_2 r_0)^i + r_0^{2i}\chi/n^2)$, where c_1, c_2 are some suitable constants independent of r. Therefore $|\Xi| = O(r^{1+\varepsilon} + \chi r^2/n^2)$, for any $\varepsilon > 0$, or $O^*(r) + O(\chi r^2/n^2)$, provided r_0 is chosen sufficiently large, and the run time is $O(n^{1+\varepsilon} + \chi r/n) = O^*(n) + O(\chi r/n)$ [34] (see also [16]).

Returning to the problem of computing a biclique partition of $\Psi\Phi P$, we follow the same overall algorithm as described in Section 2.1, except that we compute a biclique partition in Step (ii) as follows. Let v be a node of the segment tree T, let W_v be the vertical slab associated with v, and let Ψ_v, P_v be the subset of pseudo-trapezoids (clipped within W_v) and points stored at v, respectively. Let Γ_v be the set of top and bottom arcs in the pseudo-trapezoids of Ψ_v. Recall that the endpoints of Γ_v, and thus the vertical edges of Ψ_v, lie on the boundary lines of W_v. Set $N_v = |\Psi_v|$, $m_v = |P_v|$, and χ_v the number of intersection points between the arcs of Γ_v. We compute a biclique partition $\mathcal{B}_v$ of $\Psi_v\Phi P_v$, as follows.

Fix a parameter $r > 1$, whose precise value will be set later, and construct a hierarchical $(1/r)$-cutting $\Xi_0 = \mathbb{R}^2, \Xi_1, \ldots, \Xi_\nu = \Xi$ of Γ_v of size $O^*(r) + O(\chi r^2/N_v^2)$, in time $O^*(N_v) + O(\chi_v r/N_v)$. For every $i \le \nu$ and for every cell $\tau \in \Xi_i$, let Ψ_τ be the set of pseudo-trapezoids ψ whose boundary crosses τ, i.e., the top or bottom edge of ψ crosses τ. Let $\tau' \in \Xi_{i-1}$ be the parent cell that contains τ. We set $\mathcal{C}_\tau = \{\psi \in \Psi_{\tau'} \mid \tau \subseteq \psi\}$ to be the set of input pseudo-trapezoids of $\Psi_{\tau'}$ that contain τ. Set $P_\tau = P \cap \tau$. Set $N_\tau = |\Psi_\tau|$ and $m_\tau = |P_\tau|$. Finally, for each cell $\tau \in \Xi$, we compute a biclique partition $\mathcal{B}_\tau$ of $\Psi_\tau \,\Phi\, P_\tau$ using the algorithm described in the previous section (cf. Lemma 5). We set

$$\mathcal{B}_v = \{(\mathcal{C}_\tau, P_\tau) \mid \tau \in \Xi_i, 1 \le i \le \nu\} \cup \bigcup_{\tau \in \Xi} \mathcal{B}_\tau, \tag{4}$$

We repeat this step for all nodes of the segment tree and return $\bigcup_v \mathcal{B}_v$ as the desired biclique partition of $\Psi \Phi P$. Following an argument similar to that in Lemma 3, we can argue that $\mathcal{B}_v$ is indeed a biclique partition of $\Psi \,\Phi\, P$. We remark that the points of P lying on the boundary of cells in the cuttings need to be handled carefully to ensure that $\mathcal{B}_v$ is a biclique partition, but we omit the relatively straightforward details from this version.

We now analyze the size of $\mathcal{B}_v$ and the running time of the algorithm. Since r_0 is a constant and we have already computed conflict lists for each cell τ, we get that $\Psi_\tau, \mathcal{C}_\tau, P_\tau$, for all cells τ over all cuttings, can be computed in $O^*(N_v) + O(m_v \log r + \chi_v r/N_v)$ time. By Lemma 5, computing $\mathcal{B}_\tau$ takes $O^*(m_\tau N_\tau^{1/2} + N_\tau)$ expected time. Since $N_\tau = N_v/r$ and $\sum_\tau m_\tau = m_v$, the total time spent in computing $\mathcal{B}_\tau$ over all cells of Ξ is

$$\sum_{\tau \in \Xi} O^*(m_\tau N_\tau^{1/2} + N_\tau) = O^*\left(\frac{N_v^{1/2}}{r^{1/2}} \sum_{\tau \in \Xi} m_\tau + \frac{N_v}{r}|\Xi|\right) = O^*\left(\frac{N_v^{1/2}}{r^{1/2}} m_v + \chi_v \frac{r}{N_v} + N_v\right).$$

By choosing $r = \max\left\{N_v, \left\lceil N_v m_v^{2/3}/\chi_v^{2/3} \right\rceil\right\}$, the expected running time is $O^*(m_v^{2/3}\chi_v^{1/3} + m_v + N_v)$. This also bounds the size of $\bigcup_{\tau \in \Xi} |\mathcal{B}_\tau|$.

To bound the size of the first term in (4), we observe that $\sum_\tau |P_\tau|$, where the sum is taken over all cuttings Ξ_i, is $O(m_v \log r) = O^*(m_v)$. Similarly,

$$\sum_\tau |\mathcal{C}(\tau)| = O^*(N_v) + O(\chi_v r/N_v) = O^*(m_v^{2/3}\chi_v^{1/3} + N_v).$$

Hence, the total size of $\mathcal{B}_v$ is $O^*(m_v^{2/3}\chi_v^{1/3} + m_v + N_v)$.

Summing the above bound over all nodes v of T and plugging the values $\sum_v m_v = O(m \log n)$, $\sum_v N_v = O(N \log n)$, $\sum_v \chi_v \le \chi = O(n^2)$, and $N = O^*(n^{3/2})$, the expected running time, as well as the size of $\mathcal{B}$, are $O^*(m^{2/3}n^{2/3} + m + n^{3/2})$. Putting everything together, we obtain the following lemma.

▶ **Lemma 7.** *Let P be a set of m points in $\mathbb{R}^2$, and let Σ be a set of n semi-algebraic sets of constant complexity in $\mathbb{R}^2$. A biclique partition of $\Sigma \,\Phi\, P$ of size $O^*(m^{2/3}n^{2/3} + m + n^{3/2})$ can be computed in expected time $O^*(m^{2/3}n^{2/3} + m + n^{3/2})$. If χ is the number of intersection points between the edges of Σ, then the size and the expected running time are bounded by $O^*(m^{2/3}\chi^{1/3} + m + n^{3/2})$.*

4 Bicliques in Query Space: The Final Step

A weakness of the above algorithm is that the $n^{3/2}$ term in the bounds on the size and the running time dominates for $m < n^{5/4}$. To mitigate the effect of this term for smaller values of m, we apply a divide-and-conquer technique in the s-dimensional parametric space of the

query regions, so that the number of query regions reduces more rapidly than the number of input points in the recursive subproblems. When we reach subproblems for which $m \geq n^{5/4}$, we switch back to the two-dimensional plane and apply Lemma 7. This process yields the improved bound promised in Theorem 1 in the introduction.

For simplicity, we assume that the regions in Σ are defined by a single polynomial inequality. Namely, there is an $(s+2)$-variate polynomial $g(\mathbf{x}, \mathbf{y}) : \mathbb{R}^2 \times \mathbb{R}^s \to \mathbb{R}$ such that each $\sigma_i \in \Sigma$ is of the form $g(\mathbf{x}, \mathbf{y}_i) \geq 0$ for some $\mathbf{y}_i \in \mathbb{R}^s$. We denote $\mathbf{y}_i$ as $\tilde{\sigma}_i$, which is a representation of σ_i as a point in $\mathbb{R}^s$. Set $\tilde{\Sigma} = \{\tilde{\sigma}_i \mid 1 \leq i \leq n\} \subset \mathbb{R}^s$. For each $p_i \in P$, we define a semi-algebraic set $\tilde{p}_i = \{\mathbf{y} \in \mathbb{R}^s \mid g(p_i, \mathbf{y}) \geq 0\}$. Set $\tilde{P} = \{\tilde{p}_i \mid 1 \leq i \leq m\}$. Clearly, $p_i \in \sigma_j$ if and only if $\tilde{\sigma}_j \in \tilde{p}_i$. Thus a biclique $(\tilde{P}_a, \tilde{\Sigma}_a)$ of $\tilde{P} \,\Phi\, \tilde{\Sigma}$ also leads to a biclique (Σ_a, P_a) of $\Sigma \,\Phi\, P$.

We use the polynomial-partitioning technique of Guth and Katz [45] for computing bicliques of $\tilde{P} \,\Phi\, \tilde{\Sigma}$. In particular, we rely on the following result by Matoušek and Patáková [49], used for constructing a partition tree for on-line semi-algebraic range searching:

▶ **Lemma 8** (Matoušek and Patáková [49]). *Let V be an algebraic variety of dimension $k \geq 1$ in $\mathbb{R}^d$ such that all of its irreducible components have dimension k as well, and the degree of every polynomial defining V is at most some parameter E. Let $S \subset V \cap \mathbb{R}^d$ be a set of n points, and let $D \gg E$ be a parameter. There exists a polynomial $g \in \mathbb{R}[x_1, \ldots, x_d]$ of degree at most $E^{d^{O(1)}} D^{1/k}$ that does not vanish identically on any of the irreducible components of V (i.e., $V \cap Z(g)$ has dimension at most $k-1$), and each cell of $V \setminus Z(g)$ contains at most n/D points of S. Assuming D, E, d are constants, the polynomial g, a semi-algebraic representation of the cells in $V \setminus Z(g)$, and the points of S lying in each cell, can be computed in $O(n)$ time.*

Algorithm

We now describe the algorithm for computing the biclique partition. A complication in using Lemma 8 is that it does not provide any guarantees on the partitioning of the points that lie on $Z(g)$. As such, we have to handle $S \cap Z(g)$ separately. Nevertheless, the lemma does provide us with the means of doing this, as it is formulated in terms of point sets lying on a variety of any dimension. This leads to two different threads of recursion – one of them recurses on subproblems of smaller size, as in the earlier algorithms, and the other recurses on the dimension of the variety that contains the point set. We will view each recursive subproblem as associated with a node v of the recursion tree, which will naturally be a multi-level structure. Each recursive subproblem, at some node v, consists of a triple $(\mathcal{F}_v, \Sigma_v, P_v)$, where $\mathcal{F}_v$ is a set of $O(1)$ s-variate polynomials of constant degree in $\mathbb{R}[\mathbf{y}]$, and $\Sigma_v \subseteq \Sigma$ is a set of regions such that $\tilde{\Sigma}_v \subset Z(\mathcal{F}_v)$, where $Z(\mathcal{F}_v) = \bigcap_{F \in \mathcal{F}_v} Z(F)$ is the common zero set of $\mathcal{F}_v$, and $P_v \subseteq P$. Initially, $\mathcal{F}_v = \emptyset$ and $Z(\mathcal{F}_v) = \mathbb{R}^s$, $\Sigma_v = \Sigma$, and $P_v = P$. The goal is to compute a biclique partition $\mathcal{B}_v$ of $\Sigma_v \,\Phi\, P_v$, in a recursive manner.

For lack of space, we delegate the rest of this section, which presents the analysis of the size of the resulting biclique partition, and which extends the analysis to more general containment predicates, that involve more than one polynomial inequality, to the full version [11].

References

1 P. Afshani and P. Cheng, Lower bounds for semialgebraic range searching and stabbing problems, *Proc. 37th Intl. Sympos. Comput. Geom.*, pages 8:1–8:15, 2021.

2 P. Afshani and P. Cheng, On semialgebraic range reporting, *Proc. 38th Intl. Sympos. Comput. Geom.*, pages 3:1–3:14, 2022.

3 P. K. Agarwal, Simplex range searching, in *Journey Through Discrete Mathematics* (M. Loebl, J. Nešetřil and R. Thomas, eds.), Springer Verlag, Heidelberg, 2017, pp. 1–30.

4 P. K. Agarwal, N. Alon, B. Aronov, and S. Suri, Can visibility graphs be represented compactly?, *Discrete Comput. Geom.,* 12 (1994), 347–365.

5 P. K. Agarwal, B. Aronov, E. Ezra, M. J. Katz, and M. Sharir, Intersection queries for flat semi-algebraic objects in three dimensions and related problems, *Proc. 38th Intl. Sympos. Comput. Geom.*, pages 4:1–4:14, 2022. (Full version appeared as *Arxiv:2203.10241.*)

6 P. K. Agarwal, B. Aronov, E. Ezra, and J. Zahl, An efficient algorithm for generalized polynomial partitioning and its applications, *SIAM J. Comput.* 50 (2021), 760–787.

7 P. K. Agarwal, Boris Aronov, and Micha Sharir, Computing envelopes in four dimensions with applications, *SIAM J. Comput.* 26 (1997), 1714–1732.

8 P. K. Agarwal, B. Aronov, and M. Sharir, On the complexity of many faces in arrangements of pseudo-segments and of circles, in *Discrete and Computational Geometry: The Goodman-Pollack Festschrift* (B. Aronov, S. Basu, J. Pach, and M. Sharir, eds.), Springer Verlag, Berlin, 2003, pp. 1–24.

9 P. K. Agarwal, A. Efrat, and M. Sharir, Vertical decomposition of shallow levels in 3-dimensional arrangements and its applications, *SIAM J. Comput.* 29(3) (1999), 912–953.

10 P. K. Agarwal and J. Erickson, Geometric range searching and its relatives, In *Advances in Discrete and Computational Geometry*, volume 223 of *Contemp. Math.*, pages 1–56. AMS Press, Providence, RI, 1999.

11 P. K. Agarwal, E. Ezra, M. and M. Sharir, Semi-algebraic off-line range searching and biclique partitions in the plane, *Arxiv 2403.12276*, 2024.

12 P. K. Agarwal, M. J. Katz, Micha Sharir, On reverse shortest paths in geometric proximity graphs. *Comput. Geom.* 117:102053 (2024).

13 P. K. Agarwal, J. Matoušek, and M. Sharir. On range searching with semialgebraic sets II. *SIAM J. Comput.*, 42 (2013), 2039–2062.

14 P. K. Agarwal, E. Nevo, J. Pach, R. Pinchasi, M. Sharir and S. Smorodinsky, Lenses in arrangements of pseudo-circles and their applications, *J. ACM* 51(2) (2004), 139–186.

15 P. K. Agarwal and M. Sharir: Efficient randomized algorithms for some geometric optimization problems, *Discrete Comput. Geom.* 16 (1996), 317–337.

16 P. K. Agarwal and M. Sharir, Pseudoline arrangements: Duality, algorithms and applications, *SIAM J. Comput.* 34 (2005), 526–552.

17 P. K. Agarwal and K. Varadarajan, Efficient algorithms for approximating polygonal chains, *Discrete Comput. Geom.* 23 (2000), 273-–291.

18 R. Apfelbaum and M. Sharir, Large bipartite graphs in incidence graphs of points and hyperplanes, *SIAM J. Discrete Math.* 21 (2007), 707-–725.

19 B. Aronov, E. Ezra and J. Zahl, Constructive polynomial partitioning for algebraic curves in $\mathbb{R}^3$ with applications, *SIAM J. Comput.* 49 (2020), 1109–1127.

20 B. Aronov and M. Sharir, Cutting circles into pseudo-segments and improved bounds for incidences, *Discrete Comput. Geom.* 28 (2002), 475–490.

21 B. Aronov and M. Sharir, Almost tight bounds for eliminating depth cycles in three dimensions, *Discrete Comput. Geom.* 59 (2018), 725–741. Also in *Arxiv 1512.00358.*

22 S. Barone and S. Basu, Refined bounds on the number of connected components of sign conditions on a variety, *Discrete Comput. Geom.* 47 (2012), 577–597.

23 S. Basu, R. Pollack, and M.-F. Roy, *Algorithms in Real Algebraic Geometry*, 2nd Edition, Springer Verlag, Berlin, 2006.

24 M. de Berg, O. Cheong, M. J. van Kreveld, and M. H. Overmars, *Computational Geometry: Algorithms and Applications*, 3rd Edition, Springer Verlag, Berlin, 2008.

25 M. de Berg, K. Dobrindt, and O. Schwarzkopf, On lazy randomized incremental construction, *Discrete Comput. Geom.* 14 (1995), 261–286.

26 M. de Berg and O. Schwarzkopf, Cuttings and applications, *Internat. J. Comput. Geom. Appls* 5 (1995), 343–355.

27 P. Brass and Ch. Knauer, On counting point-hyperplane incidences, *Comput. Geom. Theory Appls.* 25 (2003), 13–20.

28 H. Brönnimann and M. T. Goodrich, Almost optimal set covers in finite VC-dimension, *Discrete Comput. Geom.* 14 (1995), 463–479.

29 S. Cabello, S.-W. Cheng, O. Cheong, and C. Knauer, Geometric matching and bottleneck problems, *Proc. 40th Intl. Sympos. Comput. Geom*, 2024, 31:1–31:15. Also in *Arxiv 2310.02637*, 2023.

30 P. B. Callahan and S. Rao Kosaraju, A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields, *J. ACM* 42 (1995), 67–90.

31 P. B. Callahan and S. Rao Kosaraju, Faster algorithms for some geometric graph problems in higher dimensions, *Proc. 4th Annual ACM-SIAM Sympos. Discrete Algorithms*, 1993, 291–300.

32 J. Carrier, L. Greengard, and V. Rokhlin, A fast adaptive multipole algorithm for particle simulations, *SIAM J. Sci. Stat. Comput.* 9 (1988), 669–686.

33 T. M. Chan, P. Cheng and D. W. Zheng, Semialgebraic range stabbing, ray shooting, and intersection counting in the plane, *Proc. 40th Sympos. Comput. Geom.*, 2024, 33:1–33:15.

34 B. Chazelle, Cutting hyperplanes for divide-and-conquer, *Discrete Comput. Geom.* 9 (1993), 145–158.

35 B. Chazelle, H. Edelsbrunner, L. Guibas and M. Sharir, Algorithms for bichromatic line-segment problems and polyhedral terrains, *Algorithmica* 11 (1994), 116–132.

36 F. R. K. Chung, P. Erdős, and J. Spencer, On the decomposition of graphs into complete bipartite subgraphs, in *Studies in Mathematics: To the Memory of Paul Turán*, (P. Erdős, L. Alṕar, G. Haĺasz, and A. Sárközy, eds.), Birkhäuser, Basel, 1983, pp. 95–101.

37 K. L. Clarkson, H. Edelsbrunner, L. J. Guibas, M. Sharir and E. Welzl, Combinatorial complexity bounds for arrangements of curves and spheres, *Discrete Comput. Geom.* 5 (1990), 99–160.

38 K. L. Clarkson and Peter W. Shor, Application of random sampling in computational geometry, II. *Discrete Comput. Geom.* 4 (1989), 387–421.

39 T. Do, Representation complexities of semialgebraic graphs, *SIAM J. Discrete Math.* 33 (2019), 1864–1877.

40 T. Feder and R. Motwani, Clique partitions, graph compression, and speeding-up algorithms, *J. Comput. System Sci.* 51 (1995), 261–272.

41 J. Fox, J. Pach, A. Sheffer, A. Suk, and J. Zahl, A semi-algebraic version of Zarankiewicz's problem, *J. Eur. Math. Soc.* 19 (2017), 1785–1810.

42 J. Fox, J. Pach, and A. Suk, A polynomial regularity lemma for semialgebraic hypergraphs and its applications in geometry and property testing, *SIAM J. Comput.* 45 (2016), 2199–2223.

43 J. Fox, J. Pach, and A. Suk, Density and regularity theorems for semi-algebraic hypergraphs, *Proc. 26th ACM-SIAM Sympos. Discrete Algorithms*, 2015, 1517–1530.

44 J. E. Goodman, Proof of a conjecture of Burr, Grünbaum, and Sloane, *Discrete Math.* 32 (1980), 27–35.

45 L. Guth and N. H. Katz, On the Erdős distinct distances problem in the plane, *Annals Math.* 181 (2015), 155–190.

46 M. J. Katz and M. Sharir, An expander-based approach to geometric optimization, *SIAM J. Comput.* 26 (1997), 1384–1408.

47 J. Matoušek, Geometric range searching, *ACM Comput. Surv.* 26(4) (1994), 421–461.

48 J. Matoušek, Randomized optimal algorithm for slope selection, *Inf. Process. Lett.* 39(4) (1991), 183–187.

49 J. Matoušek and Z. Patáková, Multilevel polynomial partitioning and simplified range searching, *Discrete Comput. Geom.* 54 (2015), 22–41.

50 J. Pach and M. Sharir, On the number of incidences between points and curves, *Combinat. Probab. Comput.* 7 (1998), 121–127.

51 J. Pach and M. Sharir, Geometric incidences, in *Towards a Theory of Geometric Graphs* (J. Pach, editor), Contemp. Math. 342, AMS Press, Providence, 2004, pp. 185–223.

52 M. Sharir and P. K. Agarwal, *Davenport-Schinzel Sequences and Their Geometric Applications,* Cambridge University Press, Cambridge-New York-Melbourne, 1995.

53 M. Sharir and J. Zahl, Cutting algebraic curves into pseudo-segments and applications, *J. Combinat. Theory* Ser. A 150 (2017), 1–35.

54 A. Sheffer, *Polynomial Methods and Incidence Theory*, Cambridge University Press, Cambridge, 2022.

55 L. Székely, Crossing numbers and hard Erdős problems in discrete geometry, *Combinat. Probab. Comput.* 6 (1997), 353–358.

56 E. Szemer'edi and W. T. Trotter, Extremal problems in discrete geometry, *Combinatorica* 3 (1983), 381–392.

57 Z. Tuza, Covering of graphs by complete bipartite subgraphs; complexity of 0–1 matrices, *Combinatorica* 4 (1984), 111–116.