Vertical Decomposition of Shallow Levels in 3-Dimensional Arrangements and Its Applications *

Pankaj K. Agarwal[†] Alon Efrat[‡] Micha Sharir[§]

June 17, 1996

Abstract

Let $\mathcal{F}$ be a collection of n bivariate algebraic functions of constant maximum degree. We show that the combinatorial complexity of the vertical decomposition of the $\le k$-level of the arrangement $\mathcal{A}(\mathcal{F})$ is $O(k^{3+\varepsilon}\psi(n/k))$, for any $\varepsilon > 0$, where $\psi(r)$ is the maximum complexity of the lower envelope of a subset of at most r functions of $\mathcal{F}$. This bound is nearly optimal in the worst case, and implies the existence of shallow cuttings, in the sense of [51], of small size in arrangements of bivariate algebraic functions. We also present numerous applications of these results, including: (i) data structures for several generalized three-dimensional range searching problems; (ii) dynamic data structures for planar nearest and farthest neighbor searching under various fairly general distance functions; (iii) an improved (near-quadratic) algorithm for minimum-weight bipartite Euclidean matching in the plane; and (iv) efficient algorithms for certain geometric optimization problems in static and dynamic settings.

* Work on this paper by the first author has been supported by National Science Foundation Grant CCR-93–01259, an NYI award, and by matching funds from Xerox Corp. Work on this paper by the third author has been supported by NSF Grants CCR-91-22103 and CCR-93-11127, by a Max-Planck Research Award, and by grants from the U.S.-Israeli Binational Science Foundation, the Israel Science Fund administered by the Israeli Academy of Sciences, and the G.I.F., the German-Israeli Foundation for Scientific Research and Development.

† Department of Computer Science, Box 1029, Duke University, Durham, NC 27708-0129, USA

‡ School of Mathematical Sciences, Tel Aviv University, Tel Aviv 69978, Israel

§ School of Mathematical Sciences, Tel Aviv University, Tel Aviv 69978, Israel, and Courant Institute of Mathematical Sciences, New York University, New York, NY 10012, USA

1 Introduction

In this paper we extend the recent range-searching techniques of Matoušek [51] and of Agarwal and Matoušek [6] to arrangements of bivariate algebraic functions, and derive many applications of the new techniques. As one motivation, consider the following *dynamic nearest-neighbor-searching* problem: We dynamically maintain a set S of points in the plane, under some metric δ. At any time, we wish to answer nearest-neighbor queries, in which we specify a point q and ask for the point of S nearest to q under the metric δ. If δ is the Euclidean metric, a standard lifting transformation to 3-space (as in [30]) reduces the problem to that of dynamically maintaining the lower envelope of planes in $\mathbb{R}^3$, so that, at any time during the maintenance, we can answer efficiently queries in which we specify a point $q \in \mathbb{R}^2$ and ask for the plane attaining the lower envelope at q. Using the parametric searching technique of [4], this problem can be solved by a dynamic range-searching mechanism in which the queries are points in 3-space, and we wish to determine whether the query point lies below all the planes in the current set. The techniques of [6, 51] are based on the notion of *shallow cuttings*: Given an arrangement $\mathcal{A}$ of n planes in $\mathbb{R}^3$ and parameters $k, r \leq n$, there exists a $(1/r)$-*cutting* of the first k levels of $\mathcal{A}$ of size $O(r(1 + kr/n)^2)$, i.e., there exists a decomposition of the (union of the) cells constituting these levels into $O(r(1 + kr/n)^2)$ simplices, such that the interior of each simplex is intersected by at most n/r planes. Using the existence of such shallow cuttings, Matoušek [51] constructs an efficient static data structure for halfspace-emptiness queries in $\mathbb{R}^3$: It requires $O(n \log\log n)$ storage and $O(n \log n)$ preprocessing time, and answers a halfspace emptiness query in $O(\log^2 n)$ time. This structure was later dynamized in [6]. These techniques also apply to arrangements of hyperplanes in higher dimensions.

Although these techniques can be extended to arrangements of curves in the plane, they fail for arrangements of general surfaces in three and higher dimensions. Such an arrangement does arise, for example, in the nearest-neighbor searching problem if the underlying distance function δ is not Euclidean. A specific case of this kind, which we will face in one of our applications, is where each point $s \in S$ has an additive weight $w(s)$, and the distance from a query point q to s is defined as $|qs| + w(s)$, where $|qs|$ is the Euclidean distance between these points. To tackle such cases efficiently, we need to extend the results of [6, 51] to the case of arrangements of more general functions, and this is one of the main goals of this paper.

The main technical result that we establish is the existence of shallow cuttings of small size in arrangements of (the graphs of) low-degree algebraic bivariate functions. To obtain such cuttings, we use the technique of *vertical decomposition* of cells in arrangements. This technique, described in detail in [18], is the only known general-purpose technique for decomposing cells in such arrangements into a small number of subcells of 'constant description complexity' (that is, semialgebraic subcells, each defined by a constant number of polynomial equalities and inequalities, each of constant maximum degree). It is well known that the complexity of the vertical decomposition of a cell (2-face) C in a planar arrangement of low-degree algebraic curves is proportional to the number of vertices of C. However, this property does not hold in higher dimensions (even for an arrangement of planes in $\mathbb{R}^3$): One can construct a cell in an arrangement of n planes in $\mathbb{R}^3$ that has $O(n)$ vertices and whose vertical decomposition consists of $\Omega(n^2)$ subcells. In three dimensions, the complexity (number of subcells) of the vertical decomposition of the *entire* arrangement of n low-degree algebraic bivariate functions is shown in [18] to be close to $O(n^3)$, and is thus almost optimal (in higher dimensions, though, the bounds are considerably weaker, see [18, 38] for some partial results). No similarly-sharp bounds were known for the vertical decomposition of the cells that lie only in the first k levels of the arrangement (even for arrangements of planes). Using a standard probabilistic analysis technique due to Clarkson and Shor ([25]; see also [63]), one can easily show that the combinatorial complexity of these (undecomposed) cells is $O(k^3\psi(n/k))$, where $\psi(r)$ is the maximum combinatorial complexity of the lower envelope of any subset of at most r of the given surfaces. For low-degree algebraic bivariate functions, the results of [39, 62] imply that $\psi(n) = O(n^{2+\varepsilon})$, for any $\varepsilon > 0$,[1] but in certain favorable cases, such as the case of planes, this complexity is smaller.

The first main result of the paper, derived in Section 2, is that the combinatorial complexity of the vertical decomposition of the cells in the first k levels of the arrangement is $O(k^{3+\varepsilon}\psi(n/k))$. This bound is close to optimal in the worst case. We then apply this bound to obtain a sharp bound on the size of shallow cuttings in arrangements of bivariate functions. Specifically, we show, in Theorem 3.1, that there exists a $(1/r)$-cutting of the first k levels in an arrangement of n low-degree, algebraic, bivariate functions in $\mathbb{R}^3$, as above, whose size is $O(q^{3+\varepsilon}\psi(r/q))$, where $q = k(r/n) + 1$. This bound is almost tight in the worst case, and almost coincides with the bound given in [51] for the case of planes. The proof adapts the analysis technique of [51]. If $r = O(1)$, a $(1/r)$-cutting of this size can be computed in $O(n)$ time.

An immediate consequence of Theorem 3.1 is an efficient algorithm for the following problem: Preprocess a set $\mathcal{F}$ of n bivariate functions into a data structure, so that, for a query point w, all functions whose graphs lie below w can be reported efficiently. We present a data structure of size $O(\psi(n)n^\varepsilon)$, which can be constructed in $O(\psi(n)n^\varepsilon)$ time, so that a query can be answered in $O(\log n + \xi)$ time, where ξ is the output size. If we are interested only in determining whether w lies below the graphs of all functions of $\mathcal{F}$, the query time is $O(\log n)$. (Here we are assuming a model of computation in which various operations involving a constant number of fixed-degree algebraic functions can be performed in $O(1)$ time; see below for details.) We can also modify this structure, so as to obtain an efficient data structure for dynamically maintaining a set of bivariate algebraic functions, as above, so that we can efficiently determine whether a query point lies below all the function graphs in the current set. The modified data structure also requires $O(\psi(n)n^\varepsilon)$ storage, the cost of

[1]Throughout this paper, ε denotes an arbitrarily small positive constant, and $g(n) = O(f(n) \cdot n^\varepsilon)$ means that, for any given $\varepsilon > 0$, we can choose a constant c_ε so that $g(n) \le c_\varepsilon f(n) \cdot n^\varepsilon$. A complexity bound of the form $O(f(n) \cdot n^\varepsilon)$ for an algorithm means that, for any prespecified $\varepsilon > 0$, we can tune the algorithm so that its complexity is bounded by $c_\varepsilon f(n) \cdot n^\varepsilon$, for an appropriate constant c_ε.

an update is $O(\psi(n)/n^{1-\varepsilon})$, and a query can be answered in $O(\log n)$ time.[2] The technique for constructing these data structures adapts ideas from [6, 51].

We also obtain an efficient algorithm for constructing and searching in the first k levels of an arrangement of n bivariate functions, as above. These levels can be constructed in $O(k^{3+\varepsilon}n^{\varepsilon}\psi(n/k))$ time, in an appropriate model of computation, and can be stored into a data structure of similar size, so that, for any query point p, we can determine, in $O(\log n)$ time, whether the level of p in the arrangement is at most k, and, if so, return the level of p.

We next apply the new mechanisms to a variety of geometric problems. First, in Section 6, we derive an efficient technique for dynamically maintaining a set S of points (or more general objects) in the plane, so that we can efficiently compute the nearest neighbor (or the farthest neighbor) of a query point in the current set of points, under any 'reasonable' distance function δ (defined more precisely later), which can be fairly arbitrary, and does not have to satisfy any metric-like properties. Assume that the complexity of the lower (or upper) envelope of the functions $f_i(\mathbf{x}) = \delta(p_i, \mathbf{x})$, over all objects p_i in the current set S, is at most $g(|S|)$ (this is the same as the complexity of the nearest (or farthest) neighbor Voronoi diagram of S, if δ is a metric or a convex distance function). Then our solution requires $O(g(n)n^{\varepsilon})$ storage, $O(g(n)/n^{1-\varepsilon})$ time for each update, and $O(\log n)$ time for each nearest- (or farthest-)neighbor query. We give applications of this technique to dynamic maintenance of a bichromatic closest pair between two planar point sets, under any 'reasonable' metric, and of a minimum spanning tree of a planar point set under any L_p metric.

Another interesting application of our technique is an improved algorithm for computing minimum-weight bipartite Euclidean matching for point sets in the plane (see Section 7). That is, we are given a set of n blue points and a set of n red points in the plane, and we wish to find a matching between the blue points and the red points, which minimizes the sum of the distances between the pairs of matched points. Our solution is based on the algorithm of Vaidya [67], which requires a data structure for answering nearest-neighbor queries in a dynamic setting, where the distance to each of the maintained sites is the Euclidean distance plus some additive weight associated with the site. Using our dynamic nearest-neighbor searching technique, we can improve the running time of Vaidya's algorithm from $O(n^{2.5}\log n)$ to $O(n^{2+\varepsilon})$. We also obtain an $O(n^{7/3+\varepsilon})$ algorithm for an arbitrary 'reasonable' metric. We still do not know how to obtain similar improvements in the nonbipartite case, for which the above nearest-neighbor searching mechanism is not sufficient.

Another application of our results is to dynamic maintenance of the intersection of congruent balls in 3-space, in the strong sense that we wish to determine, after each update, whether the current intersection is empty. For this, we combine our technique with the

[2] Throughout this paper, the update-time bounds are amortized. We believe that the same bounds can be achieved in the worst case, using the known (albeit complicated) techniques of [58]. Nevertheless, for the sake of simplicity, we will stick to amortized bounds, which will not affect the applications that we study here.

variant of parametric searching recently proposed in [66]. We obtain a data structure of size $O(n^{1+\varepsilon})$, which can be updated in $O(n^{\varepsilon})$ time per insertion/deletion, and which supports 'intersection-emptiness' queries in $O(\log^4 n)$ time.

We next apply our technique, in Section 9, to another problem in geometric optimization, namely the *dynamic smallest stabbing-disk* problem: We maintain dynamically a set $\mathcal{C}$ of (possibly intersecting) simply-shaped, compact, convex sets in the plane, and we wish to compute, after each insertion or deletion of such a set, the smallest disk, or the smallest homothetic copy of any compact convex set of simple shape, that intersects all the sets in the current $\mathcal{C}$. The case where we have points instead of general convex sets was recently studied in [6]. Our solution requires $O(n^{1+\varepsilon})$ storage and preprocessing, and recomputes the smallest stabbing disk after each update in $O(n^{\varepsilon})$ time. A byproduct of our analysis, which we believe to be of independent interest, is a near-linear bound on the complexity of the farthest-neighbor Voronoi diagram of (possibly intersecting) simply-shaped, compact, convex sets in the plane, under any simply-shaped, convex distance function. The bound is linear for (possibly intersecting) line segments under the Euclidean distance.

We finally present, in Section 10, a few more applications, where most of the details are omitted. The paper concludes in Section 11, with a discussion of our results and with some open problems. The collection of applications described in this paper is by no means exhaustive, and the new techniques obtained in this paper have many additional applications. The main message of this paper, in our opinion, is that there is a real need to adapt and extend range-searching and related techniques, that were originally developed for arrangements of planes or hyperplanes, to arrangements of algebraic surfaces. This adaptation is by no means easy, and the present study achieves it only for 3-dimensional arrangements, but it enlarges significantly the scope of range-searching applications.

2 Vertical Decomposition of Levels in 3-Dimensional Arrangements

Let $\mathcal{F}$ be a collection of n bivariate functions satisfying the following conditions:[3]

(F1) Each $f \in \mathcal{F}$ is a continuous, totally-defined, algebraic function of constant maximum degree b.

(F2) The functions in $\mathcal{F}$ are in *general position*. This excludes degenerate configurations where four function graphs meet at a point, a pair of graphs are tangent to each other, etc.

With some modifications of the analysis, we can also handle the case where the functions in $\mathcal{F}$ are only partially defined, and the boundary of the domain of each function is defined by a

[3] Abusing the notation slightly, we will not distinguish between a function and its graph. We will use $\mathcal{F}$ to denote a collection of functions as well as the family of surfaces representing their graphs.

constant number of polynomial equalities and inequalities of constant maximum degree, say, b too. The simplest way of doing this is to extend each $f \in \mathcal{F}$ to a totally-defined function, by forming the Minkowski sum of the graph of f with a sufficiently narrow vertical cone, whose equation is $z = c\sqrt{x^2+y^2}$, for a sufficiently large c, and by taking the lower boundary of this sum as the graph of the extended function. The resulting extended functions are easily seen to be totally-defined continuous functions, each of whose graphs is a semialgebraic set of constant description complexity. One can then check that the analysis given below also applies to collections of such functions. Concerning the general position assumption, we refer the reader to the papers [39, 62] for more details about the definition and properties of this concept. Following arguments given in these papers, it will follow that no real loss of generality is made by assuming general position.

The *arrangement* of $\mathcal{F}$, denoted as $\mathcal{A}(\mathcal{F})$, is the subdivision of $\mathbb{R}^3$ induced by the graphs of the functions in $\mathcal{F}$ (see [63] for more details). We will refer to the three-dimensional cells of $\mathcal{A}(\mathcal{F})$ simply as the *cells* of $\mathcal{A}(\mathcal{F})$. The complexity of a cell C, denoted as $|C|$, is the number of faces of all dimensions on the boundary of C. The *level* in $\mathcal{A}(\mathcal{F})$ of a point $p = (x_p, y_p, z_p) \in \mathbb{R}^3$, denoted as $\mu(p) = \mu_{\mathcal{F}}(p)$, is defined as the number of functions $f \in \mathcal{F}$ such that $z_p > f(x_p, y_p)$. The level of all points lying on a face ϕ (of any dimension) of $\mathcal{A}(\mathcal{F})$ is the same, which we denote by $\mu(\phi)$. The *k-level* of $\mathcal{A}(\mathcal{F})$, for any $0 \le k \le n-1$, is the closure of the union of all two-dimensional faces of $\mathcal{A}(\mathcal{F})$ whose level is k; the 0-level (resp. $(n-1)$-level) is the graph of the lower (resp. upper) envelope of $\mathcal{F}$. The *$(\le k)$-level* of $\mathcal{A}(\mathcal{F})$, denoted as $\mathcal{A}_{\le k}(\mathcal{F})$, is the collection of all cells of $\mathcal{A}(\mathcal{F})$ whose level is at most k. Let

$$\psi(\mathcal{F}, k) = \sum_{C \in \mathcal{A}_{\le k}(\mathcal{F})} |C|$$

denote the combinatorial complexity of $\mathcal{A}_{\le k}(\mathcal{F})$. Let $\mathbf{F}$ denote a (possibly infinite) family of bivariate functions that satisfies (F1), and let

$$\psi(n, k) = \psi^{\mathbf{F}}(n, k)$$

be any upper bound on the quantity $\max_{\mathcal{F}} \psi(\mathcal{F}, k)$, where the maximum is taken over all collections $\mathcal{F} \subseteq \mathbf{F}$ of at most n functions (that satisfy (F1) and (F2)). To simplify the presentation, we will also use $\psi(n)$ to denote $\psi(n, 0)$, and will allow n in this notation to be any real nonnegative number.

A straightforward application of the probabilistic analysis technique of Clarkson and Shor [25] (see also [63]) implies that

$$\psi(n, k) = O(k^3 \psi(n/k)). \tag{2.1}$$

We also note that, by the results of [39, 62], we always have $\psi(n) = O(n^{2+\varepsilon})$, where the constant of proportionality depends on ε and on the maximum degree of the given surfaces. Hence, in the worst case, we have $\psi(n, k) = O(k^{1-\varepsilon} n^{2+\varepsilon})$. However, these bounds can be much smaller for certain favorable underlying families $\mathbf{F}$.

Because of technical reasons, we make the following assumption on $\mathbf{F}$.

(F3) The upper bound $\psi(n)$ on the complexity of the lower envelope (and also of the upper envelope) of any collection $\mathcal{F} \subseteq \mathbf{F}$ of n functions is of the form $\psi(n) = n^\alpha \beta(n)$, where $\alpha \leq 2$ is a constant, and $\beta(n)$ is a function such that, we have $\limsup_{n\to\infty} \beta(n)/n^\delta = 0$.

This assumption involves no real loss of generality. In fact, all known bounds for $\psi(n)$ have this form with either $\alpha = 1$ or $\alpha = 2$.

The *vertical decomposition* of a (3-dimensional) cell $C \in \mathcal{A}(\mathcal{F})$, denoted as $C^\star$, is defined in the following standard manner (see [18, 28, 63] for more details):

I. For each edge e of ∂C, we erect a z-vertical wall from e, which is the union of all maximal z-vertical segments passing through points of e and lying within (the closure of) C. The collection of these walls partitions C into subcells, each of which has a unique top facet (2-dimensional face) and a unique bottom facet, each contained in some facet of C. Every z-vertical line cuts such a subcell in a (possibly empty) interval; see Figure 1.

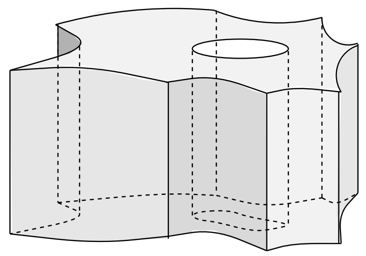

Figure 1: Vertical decomposition: A subcell created in Step I.

II. We take each of the cells Δ generated in the first step, project it onto the xy-plane, and construct the two-dimensional vertical decomposition of the projection Δ_2 of Δ, by erecting, from each vertex of Δ_2 and from each locally x-extreme point of $\partial\Delta_2$, a maximal y-vertical segment contained in the closure of Δ_2. These segments partition Δ_2 into trapezoidal-like subcells; each subcell $\tau_2 \subseteq \Delta_2$ induces a subcell τ of Δ, obtained by intersecting Δ with the vertical cylinder $\tau_2 \times \mathbb{R}$ (see Figure 2).

The cells obtained in this 2-step decomposition form the vertical decomposition of C. We define $\mathcal{A}^\star_{\leq k}(\mathcal{F})$ (resp. $\mathcal{A}^\star(\mathcal{F})$), the vertical decomposition of $\mathcal{A}_{\leq k}(\mathcal{F})$ (resp. of $\mathcal{A}(\mathcal{F})$), as the union of the vertical decompositions of all cells in $\mathcal{A}_{\leq k}(\mathcal{F})$ (resp. in $\mathcal{A}(\mathcal{F})$). The complexity of such a vertical decomposition is the total number of its cells. Notice that for each cell $\tau \in \mathcal{A}^\star(\mathcal{F})$, there is a subset $D = D(\tau)$ of at most six functions of $\mathcal{F}$, such that $\tau \in \mathcal{A}^\star(D)$. Since all the functions in $\mathcal{F}$ have constant maximum degree, it follows that each cell of $\mathcal{A}^\star(\mathcal{F})$ has *constant description complexity* [18], a property that is crucial for

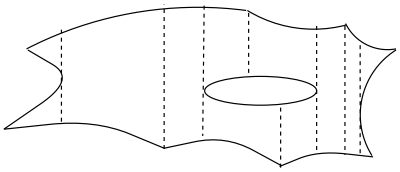

Figure 2: Vertical decomposition: Step II.

the construction of cuttings, which will be studied in the next section. We refer to the set $D(\tau)$ as the set of functions *defining* the cell τ. The main result of this section is:

Theorem 2.1 *Let $\mathcal{F}$ be a collection of n bivariate functions satisfying conditions (F1)–(F3). For any integer $k \leq n$, the combinatorial complexity of $\mathcal{A}^{\star}_{\leq k}(\mathcal{F})$ is $O(k^{3+\varepsilon}\psi(n/k))$, for any $\varepsilon > 0$, where the constant of proportionality depends on ε and on the maximum degree of the functions in $\mathcal{F}$.*

Proof: Let C be a cell of $\mathcal{A}_{\leq k}(\mathcal{F})$, and let Δ be a subcell of C created in Step I of the vertical decomposition. If $|\Delta|$ is the number of vertical edges in Δ plus the number of locally x-extremal points of edges of $\mathcal{A}_{\leq k}(\mathcal{F})$ on $\partial\Delta$, then, in Step II, Δ is partitioned into $O(|\Delta| + 1)$ subcells, because each vertical edge of Δ (resp. each locally x-extremal point as above) corresponds to a vertex of Δ_2 (resp. to a locally x-extremal point on an edge of $\partial\Delta_2$), and the number of subcells in the two-dimensional vertical decomposition of Δ_2 is easily seen to be proportional to 1 plus the number of vertices of Δ_2 plus the number of locally x-extremal points on the edges of $\partial\Delta_2$. Summing over all cells of $\mathcal{A}_{\leq k}(\mathcal{F})$, the number of subcells in $\mathcal{A}^{\star}_{\leq k}(\mathcal{F})$ is proportional to the sum of the number of vertical edges in the subcells created in Step I, the number of edges of $\mathcal{A}_{\leq k}(\mathcal{F})$, and the number of subcells created in Step I that do not contain any vertical edge or any locally x-extremal point on an edge of $\mathcal{A}_{\leq k}(\mathcal{F})$. The third quantity is bounded by $O(k^3\psi(n/k))$, because each such subcell contains at least one edge or face of $\mathcal{A}_{\leq k}(\mathcal{F})$, and each such feature of $\mathcal{A}_{\leq k}(\mathcal{F})$ can be incident to $O(1)$ subcells.

Hence, it suffices to bound the number of vertical edges in the subcells created in Step I. There are two types of vertical edges of Δ:

(V1) a z-vertical segment erected from a vertex of C, or

(V2) the intersection segment of two z-vertical walls erected from two edges e, e' of C, where e lies on the top portion of ∂C and e' lies on the bottom portion of ∂C.

Since the number of vertical edges of type (V1) is bounded by the number of vertices in C, the total number of such edges, over all cells of $\mathcal{A}_{\leq k}(\mathcal{F})$, is, as argued above, at most $O(k^3\psi(n/k))$. Hence, it suffices to bound the number of vertical edges of type (V2).

If two edges e, e' of C generate a vertical edge of type (V2), then their xy-projections must intersect. Furthermore, every intersection point between the xy-projections of the edges of ∂C, one on the upper portion and one on the bottom portion of ∂C, generates exactly one vertical edge of type (V2) (this follows from the fact that the cell C is xy-monotone). We obtain an upper bound on the number of these intersection points over all cells in $\mathcal{A}_{\leq k}(\mathcal{F})$, which in turn yields an upper bound on the number of cells in $\mathcal{A}^{\star}_{\leq k}(\mathcal{F})$.

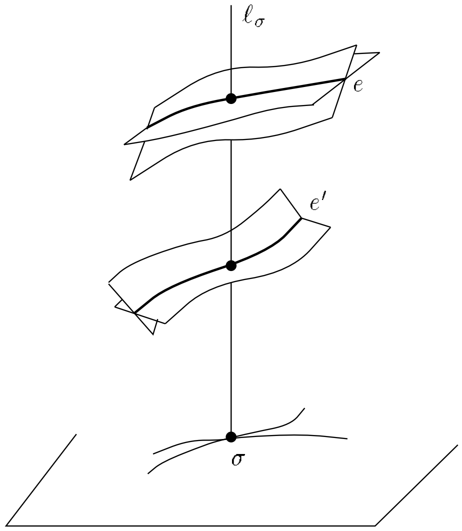

Figure 3: An edge-crossing (e, e', σ)

For a fixed pair of edges $e, e' \in \mathcal{A}(\mathcal{F})$ such that $\mu(e') < \mu(e)$, we define (e, e', σ) to be an *edge-crossing* if σ is an intersection point of the xy-projections of the (relative interiors of the) edges e and e'; see Figure 3. Let ℓ_σ denote the z-vertical line passing through σ. We define the *crossing number* of (e, e', σ) to be $\mu(e) - \mu(e') - 2$, which, by our general position assumption, is equal to the number of functions whose graphs intersect ℓ_σ strictly between e and e'. For an integer $\rho \leq \mu(e) - 2$, let $\mathcal{C}_\rho(e) = \mathcal{C}_\rho(e, \mathcal{F})$ denote the set of edge-crossings of the form (e, e', σ) whose crossing number is ρ; we put $\mathcal{C}_\rho(e, \mathcal{F}) = \emptyset$ for $\rho > \mu(e) - 2$. Set

$$\omega_\rho(e, \mathcal{F}) = |\mathcal{C}_\rho(e, \mathcal{F})| \quad \text{and} \quad \omega_{\leq \rho}(e, \mathcal{F}) = \sum_{i=0}^{\rho} \omega_i(e, \mathcal{F})\,.$$

Define

$$\varphi_\rho(\mathcal{F}, k) = \sum_{e \in \mathcal{A}_{\leq k}(\mathcal{F})} \omega_\rho(e, \mathcal{F})$$

to be the number of edge-crossings in $\mathcal{A}_{\leq k}(\mathcal{F})$ whose crossing number is ρ. Set

$$\varphi_\rho(n, k) = \max\ \varphi_\rho(\mathcal{F}, k),$$

where the maximum is taken over all subsets $\mathcal{F} \subseteq \mathbf{F}$ of size at most n. The corresponding quantities $\varphi_{\leq\rho}(\mathcal{F}, k)$ and $\varphi_{\leq\rho}(n, k)$ are defined in an obvious and analogous manner.

As follows from the above considerations, our goal is to show that

$$\varphi_0(n, k) \leq ck^{3+\varepsilon}\psi(n/k)\,, \tag{2.2}$$

for some appropriate constant c (depending on ε and on the maximum degree of the functions in $\mathbf{F}$). This will be achieved by deriving a recurrence relationship for $\varphi_0(n, k)$, whose solution will yield the desired bound. The recurrence is obtained using the following counting argument, which borrows ideas from [8].

Let e be an edge of $\mathcal{A}_{\leq k}(\mathcal{F})$, and let C be the cell of $\mathcal{A}(\mathcal{F})$ lying immediately below e. Let V_e be the vertical 2-manifold obtained as the union of all z-vertical rays emanating from the points of e in the negative z-direction. The intersection of the graph of each function $f \in \mathcal{F}$ with V_e is an algebraic arc $f^{(e)}$ of constant maximum degree, so each pair of these arcs intersect in at most a constant number of points, say, s, which depends only on the maximum degree b of the functions of $\mathcal{F}$. Let $\mathcal{A}^{(e)}(\mathcal{F})$ denote the cross-section of $\mathcal{A}(\mathcal{F})$ with V_e. See Figure 4 for an illustration. A simple but crucial observation is:

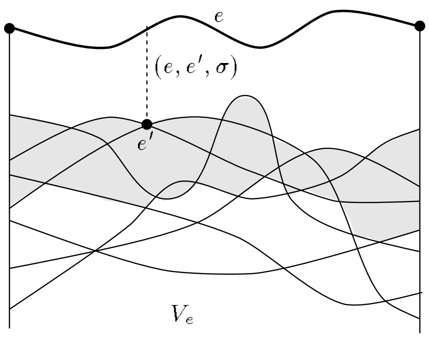

Figure 4: The arrangement $\mathcal{A}^{(e)}(\mathcal{F})$; the shaded region consists of points whose levels are between 3 and 6.

Lemma 2.2 *Let e' be an edge of $\mathcal{A}_{\leq k}(\mathcal{F})$ such that $\mu(e') < \mu(e)$. Then (e, e', σ) is an edge-crossing with crossing number ξ if and only if the intersection point $e' \cap \ell_\sigma$ is a vertex at level $\mu(e) - \xi - 2$ in $\mathcal{A}^{(e)}(\mathcal{F})$.*

The lemma implies that each edge-crossing of the form (e, e', σ) with zero crossing number corresponds to a vertex of the cross-section $C^{(e)}$ of C with V_e (which lies on the lower portion of ∂C). Let $\mathcal{F}^{(e)} \subseteq \mathcal{F}$ be the set of functions that appear on the lower portion of $\partial C^{(e)}$, and let $t_e = |\mathcal{F}^{(e)}|$. For each function $f \in \mathcal{F}^{(e)}$, we consider the cross-section $f^{(e)}$ of f within V_e, as defined above. It is easily seen that the lower portion of $\partial C^{(e)}$ is the upper envelope of the functions in $\tilde{\mathcal{F}}^{(e)} \equiv \{f^{(e)} \mid f \in \mathcal{F}^{(e)}\}$. By the standard Davenport–Schinzel theory [9, 40, 63], $\omega_0(e, \mathcal{F}) \le \lambda_s(t_e)$, where s is as above and where $\lambda_s(t)$ is the (near-linear) maximum length of a (t, s)-Davenport–Schinzel sequence.

Since the endpoints of all arcs in $\tilde{\mathcal{F}}^{(e)}$ lie on the vertical boundary of V_e and $\mu(e) \le k$, it is easily seen that $t_e \le k$, and thus $\omega_0(e, \mathcal{F}) \le \lambda_s(k)$. Summing these bounds over all edges of $\mathcal{A}_{\le k}(\mathcal{F})$, we get

$$\varphi_0(\mathcal{F}, k) = \sum_{e \in \mathcal{A}_{\le k}(\mathcal{F})} \omega_0(e, \mathcal{F}) \le \lambda_s(k)\psi(\mathcal{F}, k).$$

This yields the following weaker bound on $\varphi_0(n, k)$:

$$\varphi_0(n, k) \le \lambda_s(k) \cdot \psi(n, k) \le c_1 \lambda_s(k) k^3 \psi(n/k), \qquad (2.3)$$

for some constant c_1. The bound in (2.3) is asymptotically what we want for $k = O(1)$, and it is at most k times the desired bound for larger values of k.

We proceed now to prove the sharper bound that we are after. We fix (a sufficiently small) $\varepsilon > 0$, and choose some threshold constant k_0 (depending on ε) so that

$$4^{\varepsilon} k < c\lambda_s(k) < k^{1+\varepsilon^2/2}, \qquad (2.4)$$

for all $k > k_0$ and for some absolute constant c whose value will be determined later (the upper and lower bounds on $\lambda_s(k)$ in [9] imply that such a k_0 exists). If $k \le k_0$, then the bound asserted in Theorem 2.1 follows from (2.3), with an appropriate choice of the constant of proportionality. Henceforth, we assume that $k > k_0$. Put $q = \left\lceil (c\lambda_s(k)/k)^{1/\varepsilon} \right\rceil$. Assuming that k_0 is sufficiently large, the following inequalities, for every $k > 0$, are easy consequences of (2.4):

$$q > 4, \quad k > 4(q+1)^2, \quad \text{and} \quad k > 21(q+1)\ln k. \qquad (2.5)$$

As above, fix an edge e of $\mathcal{A}_{\le k}(\mathcal{F})$, and continue to use the notations introduced above. If $t_e \le q$ then, by Lemma 2.2, $\omega_0(e, \mathcal{F})$, which is the same as the number of vertices of $C^{(e)}$, is at most $\lambda_s(q)$. Since the number of edges in $\mathcal{A}_{\le k}(\mathcal{F})$ is at most $\psi(n, k)$, the overall number of edge-crossings with crossing number 0 and involving such edges e is at most $\lambda_s(q)\psi(n, k)$.

Next, assume that $t_e > q$. Let f, f' be a pair of distinct functions in $\mathcal{F}^{(e)}$. By continuity, f and f' must intersect within V_e at least once. Thus each function $f \in \mathcal{F}^{(e)}$ must cross at least $t_e - 1$ other functions of $\mathcal{F}$ within V_e, that is, each function $f \in \mathcal{F}^{(e)}$ is incident to at least $t_e - 1$ vertices of $\mathcal{A}^{(e)}(\mathcal{F})$. Since the graph of f contains points at level $\mu(e) - 2$ in this cross section, it follows that f is incident to at least q vertices of $\mathcal{A}^{(e)}(\mathcal{F})$ whose levels are between $\mu(e) - q - 2$ and $\mu(e) - 2$. The number of vertices of $\mathcal{A}^{(e)}(\mathcal{F})$ whose levels fall in this range is therefore $\Omega(t_e q)$, which, by the preceding analysis, implies that

$$\begin{aligned}\omega_{\le q}(e,\mathcal{F}) &= \Omega(t_e q) = \Omega\left(q t_e \cdot \frac{\omega_0(e,\mathcal{F})}{\lambda_s(t_e)}\right) \\ &\ge \frac{q}{\beta(k)} \cdot \omega_0(e,\mathcal{F}), \end{aligned} \tag{2.6}$$

where $\beta(k) = \Theta(\lambda_s(k)/k)$ is an extremely slowly growing function of k [9, 40].

Summing (2.6) over all edges e of $\mathcal{A}_{\le k}(\mathcal{F})$ for which $t_e > q$, adding the bound for the other edges of $\mathcal{A}_{\le k}(\mathcal{F})$, and observing that each edge-crossing between any two edges of $\mathcal{A}_{\le k}(\mathcal{F})$ with crossing number at most q is counted in this manner exactly once, we obtain:

$$\varphi_0(\mathcal{F},k) = \sum_{e\in\mathcal{A}_{\le k}(\mathcal{F})} \omega_0(e,\mathcal{F}) \le \frac{\beta(k)}{q}\varphi_{\le q}(\mathcal{F},k) + \lambda_s(q)\psi(n,k),$$

which implies

$$\varphi_0(n,k) \le \frac{\beta(k)}{q}\varphi_{\le q}(n,k) + \lambda_s(q)\psi(n,k). \tag{2.7}$$

In Lemma 2.3 below, we obtain the following upper bound on $\varphi_{\le q}(n,k)$:

$$\varphi_{\le q}(n,k) \le A(q+1)^4 \varphi_0\left(\left\lceil\frac{2n}{q+1}\right\rceil, \left\lceil\frac{2k}{q+1}\right\rceil\right) \tag{2.8}$$

for a constant $A > 0$. Substituting (2.1) and (2.8) in (2.7) and using the fact that $q > 4$, we obtain that

$$\varphi_0(n,k) \quad \le \quad B\left[(q+1)^3\beta(k)\varphi_0\left(\left\lceil\frac{2n}{q+1}\right\rceil, \left\lceil\frac{2k}{q+1}\right\rceil\right) + \lambda_s(q)k^3\psi\left(\frac{n}{k}\right)\right], \tag{2.9}$$

where $B > 0$ is another constant. The solution of (2.9) is

$$\varphi_0(n,k) \le Dk^{3+\varepsilon}\psi\left(\frac{n}{k}\right),$$

where $D = D(\varepsilon)$ is a sufficiently large constant depending on ε. The proof is by double induction on k and n. First, as already noted, the bound holds for $k \le k_0$, with an appropriate choice of D. For $k > k_0$ and for $n = k$, we have

$$\varphi_0(n,k) \;=\; O(n^3\beta(n)) \;=\; O(k^3\beta(k)) \;\le\; Dk^{3+\varepsilon}\psi(n/k),$$

provided D is chosen sufficiently large.

For $k > k_0$ and $n > k$, we have, by the induction hypothesis,

$$\varphi_0(n,k) \quad \le \quad B\left[(q+1)^3\beta(k)D\left\lceil\frac{2k}{q+1}\right\rceil^{3+\varepsilon}\psi\left(\left\lceil\frac{2n}{q+1}\right\rceil \Big/ \left\lceil\frac{2q}{q+1}\right\rceil\right) + \lambda_s(q)k^3\psi\left(\frac{n}{k}\right)\right].$$

Since

$$\left\lceil\frac{2n}{q+1}\right\rceil \Big/ \left\lceil\frac{2k}{q+1}\right\rceil \le \frac{2n}{k} \tag{2.10}$$

and, by condition (F3), $\psi(2n/k) = O(\psi(n/k))$, we obtain

$$\varphi_0(n,k) \le Dk^{3+\varepsilon}\psi(n/k)\left[Bc'\frac{\beta(k)}{(q+1)^\varepsilon} + \frac{B}{D}\frac{\lambda_s(q)}{k^\varepsilon}\right],$$

for an appropriate constant c'. Since $q = \left\lceil\left(\frac{c\lambda_s(k)}{k}\right)^{1/\varepsilon}\right\rceil$ and $k^{1+\varepsilon^2/2} > c\lambda_s(k)$ (see (2.4)), we have

$$(q+1)^\varepsilon > c\frac{\lambda_s(k)}{k} = \Omega(\beta(k)) \quad \text{and} \quad q \le \left\lceil k^{\varepsilon/2}\right\rceil.$$

Hence, $\lambda_s(q) = o(k^\varepsilon)$, and we thus obtain

$$Bc'\frac{\beta(k)}{(q+1)^\varepsilon} + \frac{B}{D}\frac{\lambda_s(q)}{k^\varepsilon} < 1,$$

provided that c and k_0 are chosen sufficiently large. Hence,

$$\varphi_0(n,k) \le Dk^{3+\varepsilon}\psi(n/k).$$

This establishes Theorem 2.1. □

To complete the above proof, we establish the promised lemma:

Lemma 2.3 *Let n, k, q be as defined above. There exists a constant $A > 0$ such that*

$$\varphi_{\le q}(n,k) \le A(q+1)^4\varphi_0\left(\left\lceil\frac{2n}{q+1}\right\rceil, \left\lceil\frac{2k}{q+1}\right\rceil\right).$$

Proof: We prove the lemma using a probabilistic argument that is similar to, though slightly different from, the one used by Clarkson and Shor [25]; see also [61].

Set $p = 1/(q+1)$. We choose a subset $R \subseteq \mathcal{F}$ by selecting each function of $\mathcal{F}$ independently with probability p, so the expected size of R is np. Set $k' = \lceil 2kp\rceil$, and let us bound from below the expected value of $\varphi_0(R,k')$. For an edge-crossing (e,e',σ), let $I(e,e',\sigma)$ denote the indicator function whose value is 1 if the level of $e \cap \ell_\sigma$ in $\mathcal{A}(R)$ is at most k' and $(e,e',\sigma) \in \mathcal{C}_0(e,R)$. Otherwise, $I(e,e',\sigma) = 0$. Then

$$\begin{aligned}
\mathbf{E}[\varphi_0(R,k')] &= \sum_{e\in\mathcal{A}(\mathcal{F})}\sum_{j=0}^{\mu(e)-2}\sum_{(e,e',\sigma)\in\mathcal{C}_j(e,\mathcal{F})} \Pr[\,I(e,e',\sigma) = 1\,] \\
&\ge \sum_{e\in\mathcal{A}_{\le k}(\mathcal{F})}\sum_{j=0}^{q}\sum_{(e,e',\sigma)\in\mathcal{C}_j(e,\mathcal{F})} \Pr[\,I(e,e',\sigma) = 1\,]. \qquad (2.11)
\end{aligned}$$

Fix an edge-crossing $(e,e',\sigma) \in \mathcal{C}_j(e,\mathcal{F})$, with $e \in \mathcal{A}_{\le k}(\mathcal{F})$, and $j \le q$. Let f_1, f_2 (resp. f_3, f_4) be the functions of $\mathcal{F}$ whose intersection curve contains e (resp. e'). Notice that $f_1, \ldots, f_4$ are distinct. It is easily seen that $I(e,e',\sigma) = 1$ if and only if the following three events occur simultaneously:

(E_1) $\{f_1, f_2, f_3, f_4\} \subseteq R$.

(E_2) None of the j functions of $\mathcal{F}$ whose graphs intersect ℓ_σ between e and e' is chosen in R.

(E_3) Among the $\mu(e) - j - 2$ functions whose graphs intersect ℓ_σ below e', at most $k' - 2$ are chosen in R. That is, $\mu_R(e' \cap \ell_\sigma) \le k' - 2$.

Since the subevents E_1, E_2, E_3 are independent, we have

$$\Pr[\, I(e, e', \sigma) = 1\,] = \Pr[\, E_1\,] \cdot \Pr[\, E_2\,] \cdot \Pr[\, E_3\,]\,.$$

We have $\Pr[\, E_1\,] = p^4$ and $\Pr[\, E_2\,] = (1-p)^j \ge (1-p)^q$. Therefore, it suffices to obtain a lower bound on $\Pr[E_3]$. First, observe that $\Pr[E_3] = 1$ for $\mu(e) \le k'$, so we may assume that $\mu(e) > k'$. We use the following form of Chernoff's bound to estimate $\Pr[E_3]$ (see e.g. [56, p. 425]): Let m be some integer and, for $1 \le i \le m$, let X_i be a random variable which is 1 with probability p and 0 with probability $1-p$. Assume the X_i's to be independent, and put $X = X_1 + \cdots + X_m$. Then, for any $0 < a < 2mp$, we have

$$\Pr[\, X - mp > a\,] \le \exp\left(-\frac{a^2}{4mp} + \frac{a^3}{2m^3p^3}\right). \tag{2.12}$$

If we let $f_1, \ldots, f_m$ be the functions whose graphs intersect ℓ_σ below e', and let X_i be a random variable which is 1 if $f_i \in R$ and 0 otherwise, then $\mu_R(e' \cap \ell_\sigma) = \sum_{i=1}^m X_i$. In our case, $m = \mu(e) - j - 2$, $p = 1/(q+1)$, $k' < \mu(e) \le k$, and $0 \le j \le q$. We have

$$k' - 2 = \lceil 2kp \rceil - 2 \;\ge\; 2\mu(e)p - 2 \;\ge\; 2mp - 2\,.$$

Hence,

$$\begin{aligned}
\Pr[\, \mu_R(e' \cap \ell_\sigma) > k' - 2\,] &\le \Pr[\, \mu_R(e' \cap \ell_\sigma) > 2mp - 2\,] \\
&= \Pr[\, \mu_R(e' \cap \ell_\sigma) - mp > mp - 2] \\
&\le \exp\left(-\frac{(mp-2)^2}{4mp} + \frac{(mp-2)^3}{2m^3p^3}\right).
\end{aligned}$$

Putting $x = mp$, we have

$$-\frac{(x-2)^2}{4x} + \frac{(x-2)^3}{2x^3} = -\frac{x}{4} + \frac{3}{2} - \frac{2}{x^3}(2x^2 - 3x + 2) \le -\frac{x}{4} + \frac{3}{2}\,.$$

Hence,

$$\begin{aligned}
\Pr[\, \mu_R(e' \cap \ell_\sigma) > k' - 2\,] &\le \exp\left(-\frac{mp}{4} + \frac{3}{2}\right) \\
&\le \exp\left(-\frac{\mu(e) - j - 2}{4(q+1)} + \frac{3}{2}\right)
\end{aligned}$$

$$
\begin{aligned}
&\le \exp\left(-\frac{k'-q-2}{4(q+1)}+\frac{3}{2}\right) \\
&\le \exp\left(-\frac{k}{2(q+1)^2}+\frac{7}{4}+\frac{1}{4(q+1)}\right) \\
&\le e^{-1/5},
\end{aligned}
$$

where the last inequality follows from the facts that $k > 4(q+1)^2$ and $q > 4$. This implies that $\Pr[E_3] \ge 1 - e^{-1/5}$, and therefore

$$
\Pr[\, I(e,e',\sigma) = 1\,] \ge p^4 (1-p)^q \left(1 - e^{-1/5}\right) .
$$

Since $p = \dfrac{1}{q+1}$ and $\left(1 - \dfrac{1}{q+1}\right)^q \ge \dfrac{1}{e}$, we have

$$
\Pr[\, I(e,e',\sigma) = 1\,] \ge \frac{1}{(q+1)^4}\left(\frac{1-e^{-1/5}}{e}\right) . \tag{2.13}
$$

Substituting (2.13) in (2.11),

$$
\begin{aligned}
\mathbf{E}[\, \varphi_0(R,k')\,] &\ge \sum_{e \in \mathcal{A}_{\le k}(\mathcal{F})} \sum_{j=0}^{q} \frac{1}{(q+1)^4}\left(\frac{1-e^{-1/5}}{e}\right) \cdot \varphi_j(e,\mathcal{F}) \\
&= \frac{1}{(q+1)^4}\left(\frac{1-e^{-1/5}}{e}\right) \varphi_{\le q}(\mathcal{F},k) . \qquad (2.14)
\end{aligned}
$$

On the other hand,

$$
\begin{aligned}
\mathbf{E}[\, \varphi_0(R,k')\,] &\le \Pr[\, |R| \le 2pn\,] \cdot \varphi_0(\lceil 2pn \rceil, k') + \Pr[\, |R| > 2pn\,] \cdot \varphi_0(n,k') \\
&\le \varphi_0\left(\left\lceil \frac{2n}{q+1} \right\rceil, \left\lceil \frac{2k}{q+1} \right\rceil\right) + \Pr[\, |R| > 2pn\,] \cdot \varphi_0(n,k') .
\end{aligned}
$$

By (2.3), $\varphi_0(n,k') \le c' n^5$, for some constant c', as $k' \le n$ and $\psi(n,k')$ is trivially bounded by $O(n^3)$. Since the expected size of R is pn, we obtain, using (2.12) once again,

$$
\begin{aligned}
\Pr[\, |R| > 2pn\,] \cdot \varphi_0(n,k') &\le \exp\left(-\frac{pn}{4}+\frac{1}{2}\right) \cdot c' n^5 \\
&= \exp\left(-\frac{n}{4(q+1)}+\frac{1}{2}+5\ln n + \ln c'\right) \\
&= o(1),
\end{aligned}
$$

because $k \ge 21(q+1)\ln k$, by (2.5), and $n \ge k$, which implies that $n \ge 21(q+1)\ln n$. Hence,

$$
\mathbf{E}[\, \varphi_0(R,k')\,] \le \varphi_0\left(\left\lceil \frac{2n}{q+1} \right\rceil, \left\lceil \frac{2k}{q+1} \right\rceil\right) + o(1) . \tag{2.15}
$$

Combining (2.14) and (2.15), we obtain

$$\begin{aligned}\varphi_{\leq q}(n,k) &\leq \frac{e}{1-e^{-1/5}}(q+1)^4 \cdot \left[\varphi_0\left(\left\lceil\frac{2n}{q+1}\right\rceil, \left\lceil\frac{2k}{q+1}\right\rceil\right) + o(1)\right] \\ &\leq A(q+1)^4\varphi_0\left(\left\lceil\frac{2n}{q+1}\right\rceil, \left\lceil\frac{2k}{q+1}\right\rceil\right), \qquad (2.16)\end{aligned}$$

for an appropriate constant A. This completes the proof of the lemma, and thus also of Theorem 2.1. □

3 Shallow Cuttings

In this section we first define $(1/r)$-cuttings of $\mathcal{A}_{\leq k}(\mathcal{F})$, and then prove the existence of such cuttings with small size.

Let $\mathcal{F}$ be a collection of bivariate functions satisfying conditions (F1) and (F3). We call a region (or cell) $\Delta \subseteq \mathbb{R}^3$ *primitive* if there exists a set $D = D(\Delta)$ of at most 6 functions in $\mathcal{F}$ such that $\Delta \in \mathcal{A}^\star(D)$, and *weakly-primitive* if it is the intersection of two primitive cells. For a primitive or a weakly-primitive cell Δ, let $\mathcal{F}_\Delta \subseteq \mathcal{F}$ be the set of functions whose graphs intersect the interior of Δ, and put $n_\Delta = |\mathcal{F}_\Delta|$. A set Ξ of pairwise-disjoint weakly-primitive cells is called a *$(1/r)$-cutting of $\mathcal{A}_{\leq k}(\mathcal{F})$* if the union of Ξ contains $\mathcal{A}_{\leq k}(\mathcal{F})$ and if $n_\Delta \leq n/r$ for every $\Delta \in \Xi$. We also refer to such a cutting as a *shallow cutting* in the arrangement $\mathcal{A}(\mathcal{F})$.

By the ε-net theory [41] and the result of [18], there exists a $(1/r)$-cutting of the entire $\mathcal{A}(\mathcal{F})$ that consists of $O(r^3\beta(r)\log^3 r)$ *primitive* cells, where $\beta(r)$ is the extremely slowly growing function defined in the previous section. Moreover, if r is a constant, such a cutting can be constructed in $O(n)$ deterministic time, using the technique of [49]. We now prove the existence of small-size $(1/r)$-cuttings of $\mathcal{A}_{\leq k}(\mathcal{F})$.

Theorem 3.1 *Let $\mathcal{F}$ be a collection of n bivariate functions, as above, and let $k, r < n$ be integers. Set $q = k(r/n)+1$. Then there exists a $(1/r)$-cutting Ξ of $\mathcal{A}_{\leq k}(\mathcal{F})$ whose size is at most $C_1 q^{3+\varepsilon}\psi(r/q)$, where $C_1 = C_1(\varepsilon)$ is an appropriate constant. Moreover, if $r = O(1)$, a $(1/r)$-cutting of this size can be computed in $O(n)$ time.*

Matoušek [51] proved a similar result for the case of linear functions. We follow his proof, but we need to enhance it with additional machinery.

Proof: For a subset $\mathcal{G} \subseteq \mathcal{F}$ and an integer $j < n$, let $\mathcal{T}_j(\mathcal{G})$ be the set of primitive cells Δ in $\mathcal{A}^\star(\mathcal{G})$ that lie completely in $\mathcal{A}_{\leq j}(\mathcal{F})$ (i.e., the level of all points in Δ with respect to $\mathcal{F}$ is at most j). In other words,

$$\mathcal{T}_j(\mathcal{G}) = \{\Delta \in \mathcal{A}^\star(\mathcal{G}) \mid \mu_{\mathcal{F}}(p) \leq j \text{ for all } p \in \Delta\}.$$

Let $\mathcal{T}_j = \bigcup_{\mathcal{G} \subseteq \mathcal{F}} \mathcal{T}_j(\mathcal{G})$.

Fix $p = r/n$. Choose a subset $R \subseteq \mathcal{F}$ by selecting each function of $\mathcal{F}$ independently, with probability p. For each cell $\Delta \in \mathcal{A}^\star(R)$ that has a point whose level with respect to $\mathcal{F}$ is at most k, we do the following. If $|n_\Delta| \le n/r$, we add Δ to Ξ. Otherwise, suppose that $(t-1)n/r < n_\Delta \le tn/r$, for some integer $t > 1$ (we then say that the *excess* of such a Δ is t). We compute a $(1/t)$-cutting Ξ_Δ of $\mathcal{A}(\mathcal{F}_\Delta)$, which consists of $O(t^3\beta(t)\log^3 t)$ primitive cells, clip each cell of Ξ_Δ within Δ and add the resulting (weakly-primitive) cells to Ξ. This yields a $(1/r)$-cutting Ξ of $\mathcal{A}_{\le k}(\mathcal{F})$. We now show that the expected size of Ξ is as asserted in the theorem.

Let $\eta(p,t,j)$ denote the expected number of those cells Δ in $\mathcal{T}_j(R)$ whose excess is at least t. If any cell $\Delta \in \mathcal{A}^\star(R)$ with excess t intersects $\mathcal{A}_{\le k}(\mathcal{F})$, then the level of all points of Δ with respect to $\mathcal{F}$ is at most $k + tn/r$. This is easily seen to imply

$$\mathbf{E}[|\Xi|] \;\le\; \eta(p,0,k) + \sum_{t\ge 1} O(t^3\beta(t)\log^3 t)\cdot \eta\left(p,t,k+\left\lfloor \frac{tn}{r}\right\rfloor\right).$$

A primitive cell $\Delta \in \mathcal{T}_j$ appears in $\mathcal{T}_j(R)$ if and only if $D(\Delta) \subseteq R$ and $\mathcal{F}_\Delta \cap R = \emptyset$. Using the same argument as in [51] (see also [7, 20]), one can show that for $t \ge 1$,

$$\eta(p,t,j) \le c2^{-t}\cdot \eta\left(\frac{p}{t},0,j\right) \tag{3.1}$$

for a constant $c > 0$. We prove in Lemma 3.2 below that for any $0 < p < 1$ and for any integer $l < n$,

$$\eta\,(p,0,l) = O\left((1+lp)^{3+\varepsilon}\psi\,(n/l)\right).$$

Hence,

$$\begin{aligned}\mathbf{E}[|\Xi|] &= O\left((1+kp)^{3+\varepsilon}\psi\,(n/k)\right) + \\ &\quad \sum_{t\ge 1} O(t^3 2^{-t}\beta(t)\log^3 t)\cdot\left(1+\frac{p(k+\lfloor tn/r\rfloor)}{t}\right)^{3+\varepsilon}\psi\left(\frac{n}{k+\lfloor tn/r\rfloor}\right)\\ &= O\Big((1+kp)^{3+\varepsilon}\psi(n/k)\Big) + O\left(q^{3+\varepsilon}\psi(r/q)\right)\sum_{t\ge 1}\frac{t^3\beta(t)\log^3 t}{2^t}.\end{aligned}$$

Condition (F3) allows us to write

$$\begin{aligned}(1+kp)^{3+\varepsilon}\psi\left(\frac{n}{k}\right) &= \left(1+\frac{kr}{n}\right)^{3+\varepsilon}\psi\left(\frac{n}{k}\right)\\ &= O\left(q^{3+\varepsilon}\psi\left(\frac{n}{k+n/r}\right)\right)\\ &= O\left(q^{3+\varepsilon}\psi\left(\frac{r}{q}\right)\right).\end{aligned}$$

Hence,

$$\mathbf{E}[|\Xi|] = O\left(q^{3+\varepsilon}\psi\left(\frac{r}{q}\right)\right).$$

Finally, observe that the proof of this theorem is constructive, so we immediately obtain a randomized algorithm for computing Ξ. If $r = O(1)$, then the expected running time is $O(n)$. The algorithm can be made deterministic using the method of conditional probabilities, as described in [16, 49]. $\square$

To complete the analysis, we establish the promised lemma:

Lemma 3.2 *For any real number $0 < p < 1$ and for any integer $k < n$,*

$$\eta(p, 0, k) = O\left((1 + kp)^{3+\varepsilon}\psi\left(\frac{n}{k}\right)\right).$$

Proof: Let R be, as above, a random subset of $\mathcal{F}$ obtained by choosing each element of $\mathcal{F}$ independently, with probability p. It suffices to bound the expected number of those primitive cells Δ in $\mathcal{A}^\star(R)$ that belong to $\mathcal{T}_k(R)$, i.e., the level of all points in Δ with respect to $\mathcal{F}$ is at most k. Each cell $\Delta \in \mathcal{A}^\star(R)$ is of one of the following three types (these types are not mutually exclusive):

(C1) Δ contains a vertex v of $\mathcal{A}(R)$; if $\Delta \in \mathcal{T}_k(R)$ then $\mu_{\mathcal{F}}(v) \le k$.

(C2) Δ has a vertical face parallel to the yz-plane and tangent to an edge e of $\mathcal{A}(R)$ at some point w; if $\Delta \in \mathcal{T}_k(R)$ then $\mu_{\mathcal{F}}(w) \le k$.

(C3) Δ has a vertical edge induced by an edge-crossing $(e, e', \sigma) \in \mathcal{C}_0(e, R)$ for some $e \in \mathcal{A}(R)$. If $\Delta \in \mathcal{T}_k(R)$ then $\mu_{\mathcal{F}}(e \cap \ell_\sigma) \le k$.

A vertex $v \in \mathcal{A}_{\le k}(\mathcal{F})$ appears as a vertex of $\mathcal{A}(R)$ if and only if the three functions whose graphs contain v are chosen in R. Moreover, under the general position assumption, each vertex of $\mathcal{A}(R)$ appears in $O(1)$ cells of $\mathcal{A}^\star(R)$. Therefore, the expected number of cells in $\mathcal{T}_k(R)$ of type (C1) is at most

$$O\Bigg(\sum_{v \in \mathcal{A}_{\le k}(\mathcal{F})} \Pr[\, v \text{ is a vertex of } \mathcal{A}(R)\,]\Bigg) = O(p^3\psi(n, k)) = O((kp)^3)\psi\left(\frac{n}{k}\right).$$

Similarly, the number of cells in $\mathcal{T}_k(R)$ of type (C2) can be bounded by

$$O\Bigg(\sum_w \Pr[\, w \text{ appears in } \mathcal{A}(R)\,]\Bigg),$$

where the sum is taken over all locally x-extremal points on the edges of $\mathcal{A}_{\le k}(\mathcal{F})$. The probability of such a point appearing in $\mathcal{A}_{\le k}(\mathcal{F})$ is p^2. Another application of the Clarkson-Shor technique [25] implies that the number of points w in the above sum is $O(k^2\psi'(n/k))$, where $\psi'(n/k)$ is the number of such points on the lower envelope of at most n/k functions of $\mathcal{F}$. Since $\psi'(n/k) \le \psi(n/k)$, we can bound the above sum by

$$O\left((kp)^2\psi\left(\frac{n}{k}\right)\right) = O\left((1+kp)^3\psi\left(\frac{n}{k}\right)\right) .$$

Next, consider the cells of type (C3). Let

$$\Phi(R,k) = \bigcup_{e\in\mathcal{A}(R)} \{(e,e',\sigma)\in\mathcal{C}_0(e,R) \mid \mu_{\mathcal{F}}(e\cap\ell_\sigma)\le k\} .$$

In order to bound the expected number of cells of type (C3), it suffices to obtain an upper bound on the expected value of $|\Phi(R,k)|$.

We proceed as in the proof of Lemma 2.3. Let e be an edge of $\mathcal{A}_{\le k}(\mathcal{F})$, and let $(e,e',\sigma)\in\mathcal{C}_j(e,\mathcal{F})$ for some $0\le j\le k-2$. The edge-crossing (e,e',σ) contributes (exactly) one edge-crossing to $\Phi(R,k)$ if and only if the four functions of $\mathcal{F}$ defining e and e' are chosen in R and none of the j functions whose graphs intersect the line ℓ_σ between e and e' is chosen in R. The probability that (e,e',σ) gives rise to one edge-crossing in $\Phi(R,k)$ is thus $p^4(1-p)^j$. Since every edge-crossing of $\Phi(R,k)$ is of this form, we obtain

$$\begin{aligned}
\mathbf{E}[|\Phi(R,k)|] &= \sum_{j=0}^{k-2}\sum_{e\in\mathcal{A}_{\le k}(\mathcal{F})}\sum_{(e,e',\sigma)\in\mathcal{C}_j(e,\mathcal{F})} p^4(1-p)^j \\
&= \sum_{j=0}^{k-2} p^4(1-p)^j\varphi_j(\mathcal{F},k) \\
&= p^4\varphi_0(\mathcal{F},k) + \sum_{j=1}^{k-2} p^4(1-p)^j\left(\varphi_{\le j}(\mathcal{F},k) - \varphi_{\le j-1}(\mathcal{F},k)\right) \\
&\le p^4(1-p)^{k-2}\varphi_{\le k-2}(n,k) + \sum_{j=0}^{k-3} p^5(1-p)^j\varphi_{\le j}(n,k) \\
&\le p^4(1-p)^{k-2}\cdot A(k-1)^4\varphi_0\left(\left\lceil\frac{2n}{k-1}\right\rceil, \left\lceil\frac{2k}{k-1}\right\rceil\right) + \\
&\quad \sum_{j=0}^{k-3} p^5(1-p)^j\cdot A(j+1)^4\varphi_0\left(\left\lceil\frac{2n}{j+1}\right\rceil, \left\lceil\frac{2k}{j+1}\right\rceil\right) \\
&\qquad \text{(Applying eq. (2.8))} \\
&\le Ap^4(1-p)^{k-2}(k-1)^4\cdot D\left\lceil\frac{2k}{k-1}\right\rceil^{3+\varepsilon}\psi\left(\frac{2n}{k}\right) + \\
&\quad \sum_{j=0}^{k-3} Ap^5(1-p)^j(j+1)^4\cdot D\left\lceil\frac{2k}{j+1}\right\rceil^{3+\varepsilon}\psi\left(\frac{2n}{k}\right) \\
&\qquad \text{(Applying eqs. (2.9) and (2.10))}
\end{aligned}$$

$$= O\left((1+kp)^{3+\varepsilon}\psi\left(\frac{n}{k}\right)\right)\left[p(k-1)(1-p)^{k-2} + \sum_{j=0}^{k-3} p^2(j+1)(1-p)^j\right]$$
$$= O\left((1+kp)^{3+\varepsilon}\psi\left(\frac{n}{k}\right)\right).$$

The last inequality follows from the fact that the sum in the square brackets is $1-(1-p)^{k-1} \leq 1$. This completes the proof of the lemma, and thus of Theorem 3.1. □

Remark 3.3 We note that if $\mathcal{F}$ is a collection of n univariate functions, then $\mathcal{A}^\star(\mathcal{F})$ has only $O(n^2)$ cells, every cell in $\mathcal{A}^\star(\mathcal{F})$ is of type (C1) or (the appropriate variant of type) (C2), and $\psi(n,k) = O(k^2\lambda_s(n/k))$, where s is the maximum number of intersections between the graphs of any pair of functions in $\mathcal{F}$. Following the preceding analysis, one can show that there exists a $(1/r)$-cutting of $\mathcal{A}_{\leq k}(\mathcal{F})$ of size $O(q^2\lambda_s(r/q))$, where $q = k(r/n)+1$.

4 Range Searching

An immediate consequence of Theorem 3.1 is an efficient solution of the following problem: Given a set $\mathcal{F} \subseteq \mathbf{F}$ of n bivariate functions satisfying the conditions (F1) and (F3), preprocess it into a data structure so that, for a query point $p = (x_p, y_p, z_p)$, the subset $\{f \in \mathcal{F} \mid f(x_p, y_p) < z_p\}$ can be reported efficiently. We also want to update $\mathcal{F}$ dynamically by inserting and deleting functions of $\mathbf{F}$ into/from $\mathcal{F}$.

Chazelle et al. [18] have shown that $\mathcal{F}$ can be preprocessed in time $O(n^{3+\varepsilon})$ into a data structure of size $O(n^{3+\varepsilon})$ so that a query can be answered in $O(\log n + \xi)$ time, where ξ is the output size. In this section we present a data structure that uses $O(\psi(n)\cdot n^\varepsilon)$ storage and preprocessing time, answers a query in $O(\log n + \xi)$ time, and can be updated dynamically at a small (amortized) cost. We first present a static data structure, and then explain how to dynamize it.

4.1 A static data structure

We construct a tree data structure T to answer queries of the above form. Each node v of T is associated with a subset $\mathcal{F}^v \subseteq \mathcal{F}$ and a weakly-primitive cell $\Delta_v \subseteq \mathbb{R}^3$. The root is associated with $\mathcal{F}$ and with the entire $\mathbb{R}^3$. Fix a sufficiently large constant r and set $k = \lfloor 2n/r \rfloor$. If $|\mathcal{F}|$ is less than some pre-specified constant n_0, then T consists of just the root. Otherwise, we compute a $(1/r)$-cutting Ξ of $\mathcal{A}_{\leq k}(\mathcal{F})$, whose size, by Theorem 3.1 and (F3), is $O(\psi(r))$ (where the constant of proportionality is independent of r). For each cell $\Delta \in \Xi$, let $\mathcal{F}_\Delta^- \subseteq \mathcal{F}$ be the set of functions whose graphs either intersect Δ or lie below Δ. If $|\mathcal{F}_\Delta^-| > 3n/r$, we discard Δ, because Δ lies completely above $\mathcal{A}_{\leq k}(\mathcal{F})$. Otherwise, we create a child $v = v_\Delta$ of the root, corresponding to Δ, put $\Delta_v = \Delta$ and $\mathcal{F}^v = \mathcal{F}_\Delta^-$, and recursively preprocess $\mathcal{F}^v$ (in this recursive processing, we use the same parameter r, and set $k = \lfloor 2|\mathcal{F}^v|/r \rfloor$). The recursion stops when we reach nodes v with $|\mathcal{F}^v| \leq n_0$. If $S(n)$ denotes the maximum space used by the data structure, on any set $\mathcal{F}$ of n functions of $\mathbf{F}$, then we obtain the recurrence

$$S(n) \leq \begin{cases} c_0 & n \leq n_0, \\ c_1\psi(r) \cdot S(3n/r) + c_2 n & n > n_0, \end{cases}$$

where c_0, c_1, c_2 are appropriate constants (independent of r). Recall that, by (F3), the bound $\psi(n)$ can be written as $n^\alpha \beta(n)$, where $1 \leq \alpha \leq 2$ is some constant and where $\limsup_{n\to\infty} \beta(n)/n^\delta = 0$ for any $\delta > 0$. The solution of the recurrence is

$$S(n) \leq B\psi(n) \cdot n^\varepsilon,$$

for any $\varepsilon > 0$, where B depends on ε. The proof is by induction on n. The claim holds for all $n \leq n_0$, provided that B is chosen sufficiently large. For larger values of n, by the induction hypothesis,

$$\begin{aligned} S(n) &\leq c_1\psi(r) \cdot B\psi(3n/r) \cdot (3n/r)^\varepsilon + c_2 n \\ &\leq c_1 B r^\alpha \beta(r)(3n/r)^\alpha \beta(3n/r) \cdot (3n/r)^\varepsilon + c_2 n \\ &\leq B \cdot n^\alpha \beta(n) \cdot n^\varepsilon \left[\frac{3^{\alpha+\varepsilon} c_1 \beta(r)}{r^\varepsilon} + \frac{c_2 n^{1-\alpha-\varepsilon}}{B\beta(n)} \right] \\ &\leq B\psi(n) n^\varepsilon, \end{aligned}$$

provided that r and B are chosen sufficiently large (as functions of ε). The preprocessing time of the data structure is also easily seen to be $O(\psi(n) \cdot n^\varepsilon)$.

Let p be a query point. To answer the query for p, we trace a path in T, starting from the root. Suppose we are at a node v. If v is a leaf or if there is no child w of v for which $p \in \Delta_w$, we explicitly check for each $f \in \mathcal{F}^v$ (the set of functions associated with v) whether p lies above the graph of f, and we report those functions which satisfy this condition. The time spent at v is then $O(|\mathcal{F}^v|)$, but then either $|\mathcal{F}^v| \leq n_0$ or p lies above the $\lfloor 2|\mathcal{F}^v|/r \rfloor$-level in $\mathcal{A}(\mathcal{F}^v)$, so we report at least $\lfloor 2|\mathcal{F}^v|/r \rfloor$ functions. If there is a child w of v such that $p \in \Delta_w$, we recursively visit that child. The overall query time is therefore $O(\log n + \xi)$, where ξ is the output size of the query. This procedure can be modified so that it only determines whether p lies below the graphs of all the functions of $\mathcal{F}$. In particular, if v is a leaf, we explicitly check whether p lies above the graph of any of the functions in $\mathcal{F}^v$. If v is an interior node and there is a child w of v such that $p \in \Delta_w$, we recursively visit w. Otherwise, we conclude that p does not lie below the graphs of all functions in $\mathcal{F}$. This modified procedure takes $O(\log n)$ time. Hence, we obtain:

Theorem 4.1 *Given a family $\mathbf{F}$ of bivariate functions satisfying (F1) and (F3), and a set $\mathcal{F} \subseteq \mathbf{F}$ of size n, we can preprocess $\mathcal{F}$ (in an appropriate model of computation), in time $O(\psi(n) \cdot n^\varepsilon)$, into a data structure $\Pi(\mathcal{F})$ of size $O(\psi(n) \cdot n^\varepsilon)$ so that for a query point p, all ξ functions whose graphs lie below p can be reported in time $O(\log n + \xi)$. The same data structure can be used to determine in $O(\log n)$ time whether a query point lies below the graphs of all functions in the current set $\mathcal{F}$.*

4.2 A dynamic data structure

We can dynamize the above data structure, using the technique of Agarwal and Matoušek [6]. Since the basic idea is the same, we only give a brief description of the structure. For the sake of simplicity, we describe a data structure that handles only deletions; insertions can then be handled using standard decomposition techniques [6, 15].

We need the following lemma, which is an easy consequence of a result by Chazelle et al. [18], but we give a somewhat simpler proof.

Lemma 4.2 *Let Ξ be a cutting in $\mathbb{R}^3$ consisting of t cells. Ξ can be preprocessed in time $O(t^3)$ into a data structure of size $O(t^3)$, so that, for a query point p, we can determine in $O(\log t)$ time the cell of Ξ containing p, or can conclude that there is no such cell.*

Proof: Assume that the faces of Ξ are xy-monotone, which will hold in our applications (however, the proof also holds when this is not the case, by appropriately decomposing each face of the cutting cells (each of which is assumed to have constant description complexity) into a constant number of xy-monotone subfaces). Project each face onto the xy-plane. Let Ξ^* be the set of the resulting planar patches. For a point $p \in \mathbb{R}^2$, let $\Lambda(p)$ be the list of faces of Ξ, sorted in the $+z$-direction, that intersect the z-vertical line passing through p. We compute the arrangement $\mathcal{A}(\Xi^*)$. Since the interiors of all cells of Ξ are pairwise disjoint, $\Lambda(p)$ is same for all points within a face of $\mathcal{A}(\Xi^*)$. For each face $\phi \in \mathcal{A}(\Xi^*)$, we store this sorted list, denoted by $\Lambda(\phi)$. This requires a total of $O(t^3)$ space and can be computed in time $O(t^3)$. (In fact, the storage and preprocessing time can be improved to $O(t^2)$ and $O(t^2 \log t)$, respectively, using persistent data structures, but the weaker bounds suffice for our purpose.) To locate a point $w \in \mathbb{R}^3$ in Ξ, we first find the face of $\mathcal{A}(\Xi^*)$ that contains the xy-projection of w, and then search with w in the appropriate list $\Lambda(\phi)$ to determine the cell of Ξ containing w. If Ξ does not cover the entire $\mathbb{R}^3$, the search in $\Lambda(\phi)$ can also determine whether w lies outside Ξ. For example, in the case of shallow cuttings, if w lies above all faces in $\Lambda(\phi)$, we can conclude that w does not lie in any cell of Ξ. □

We are now in position to describe the data structure. The overall data structure, denoted as $\Psi(\mathcal{F})$, is a tree of constant depth. Each node v of $\Psi(\mathcal{F})$ is associated with a subset $\mathcal{F}^v \subseteq \mathcal{F}$. The subtree rooted at any node of $\Psi(\mathcal{F})$ is reconstructed periodically after performing some deletions. Let m_v (resp. m) be the size of $\mathcal{F}^v$ (resp. $\mathcal{F}$) when the structure was last reconstructed, and let n_v (resp. n) denote the size of the current set $\mathcal{F}^v$ (resp. $\mathcal{F}$). Set $r = m^\delta$, for some sufficiently small $\delta > 0$. (The value of r is the same for all nodes of the tree, and it is updated only when the entire structure $\Psi(\mathcal{F})$ is reconstructed.) We reconstruct the subtree rooted at v after performing $m_v/2r$ deletions from that subtree. Hence, $n_v \geq m_v(1 - \frac{1}{2r})$. A function $f \in \mathcal{F}$ is said to be *relevant* for a cell Δ if the graph of f either intersects Δ or lies below Δ.

If $|\mathcal{F}| \leq r$, $\Psi(\mathcal{F})$ consists of a single node at which we preprocess $\mathcal{F}$ into the range-searching data structure, $\Pi(\mathcal{F})$, provided in Theorem 4.1. Otherwise, the root u of $\Psi(\mathcal{F})$

stores the following information:

(i) A partition of $\mathcal{F}^v$ into pairwise-disjoint sets $\mathcal{F}_1, \ldots, \mathcal{F}_t$, where $t \leq \lceil 1/\delta^2 \rceil$.

(ii) A cutting Ξ_i, for each $1 \leq i \leq t$. Initially, Ξ_i is a $(1/r)$-cutting of $\mathcal{A}_{\leq k}\left(\bigcup_{j \leq i} \mathcal{F}_j\right)$, where k is a parameter defined below, and each function of $\mathcal{F}_i$ is relevant for at most

$$\kappa = 2C_1 \frac{\psi(r)}{r^{1-\delta}} \tag{4.1}$$

cells of Ξ_i, where C_1 is the constant arising in (4.2) below.

(iii) A data structure for point location queries, for each Ξ_i, as provided in Lemma 4.2.

(iv) A pointer to a subtree $\Psi(\mathcal{F}_{i,\Delta})$, for every i and for every $\Delta \in \Xi_i$, where $\mathcal{F}_{i,\Delta} \subseteq \mathcal{F}_i$ is the set of functions relevant for Δ.

(v) For each $1 \leq i \leq t$ and for every $f \in \mathcal{F}_i$, we the set $L(f)$ of cells $\Delta \in \Xi_i$ for which f is relevant.

(vi) A counter $dcount_u$, which is initially set to $m/2r$.

(vii) The range-searching data structure $\Pi_v = \Pi(\mathcal{F}^v)$ on $\mathcal{F}^v$, as provided in Theorem 4.1.

This information is computed as follows. We first construct Π_v in $O(\psi(m)m^\varepsilon)$ time, using Theorem 4.1. Next, we construct the $\mathcal{F}_i$'s and Ξ_i's. Suppose we have already computed $\mathcal{F}_1, \ldots, \mathcal{F}_{i-1}$. Let $\overline{\mathcal{F}}_i = \mathcal{F}^v - (\mathcal{F}_1 \cup \cdots \cup \mathcal{F}_{i-1})$, and let $m_i = |\overline{\mathcal{F}}_i|$. If $m_i \leq m/r$, then Ξ_i consists of just one sufficiently large primitive cell, and $\mathcal{F}_i = \overline{\mathcal{F}}_i$. Otherwise, set

$$r_i = r\frac{m_i}{m} \quad \text{and} \quad k = \left\lfloor \frac{m}{r} \right\rfloor = \left\lfloor \frac{m_i}{r_i} \right\rfloor.$$

Using Theorem 3.1, we compute a $(1/r_i)$-cutting Ξ_i for $\mathcal{A}_{\leq k}(\overline{\mathcal{F}}_i)$ whose size is at most $C_1\psi(r_i)$. For each cell $\Delta \in \Xi_i$, let $\overline{\mathcal{F}}_{i,\Delta} \subseteq \overline{\mathcal{F}}_i$ be the set of functions relevant for Δ. We have $|\overline{\mathcal{F}}_{i,\Delta}| \leq k + m_i/r_i \leq 2m_i/r_i$. Therefore

$$\sum_{\Delta \in \Xi_i} |\overline{\mathcal{F}}_{i,\Delta}| \leq C_1 \frac{2m_i}{r_i}\psi(r_i) \;=\; 2C_1 m_i \frac{\psi(r_i)}{r_i}. \tag{4.2}$$

We call a function 'good' if it is relevant for at most κ cells. Then (4.2) implies that the number of 'bad' functions is at most

$$2C_1 m_i \frac{\psi(r_i)}{r_i} \Big/ 2C_1 \frac{\psi(r)}{r^{1-\delta}} \;=\; \frac{m_i}{r^\delta} \cdot \frac{\psi(r_i)}{r_i} \Big/ \frac{\psi(r)}{r} \;\leq\; \frac{m_i}{m^{\delta^2}}.$$

Let $\mathcal{F}_i$ be the set of good functions of $\overline{\mathcal{F}}_i$. We now repeat the construction for $\overline{\mathcal{F}}_{i+1} = \overline{\mathcal{F}}_i - \mathcal{F}_i$. The above analysis implies that $|\overline{\mathcal{F}}_i| \leq m^{1-i\delta^2}$, so the process terminates after at most $\lceil 1/\delta^2 \rceil$

steps. The construction of the structure is now completed by recursively building the data structure $\Psi(\mathcal{F}_{i,\Delta})$ for each $\Delta \in \Xi_i$. (The remaining pointers, lists, and counters are trivial to produce.) The size of the data structure and the preprocessing time are both $O(\psi(n)n^{\varepsilon})$, arguing as in [6], and the choice of r ensures that the depth of $\Psi(\mathcal{F})$ is $O(1)$.

Deleting a function. We first describe how to delete a function $f \in \mathbf{F}$ from $\mathcal{F}$. We traverse $\Psi(\mathcal{F})$ in a top-down fashion. Suppose we are at a node v. We decrease the variable $dcount_v$ by 1. If $dcount_v = 0$, we reconstruct $\Psi(\mathcal{F}^v)$, including Π_v, with the current set of functions in $\mathcal{F}^v$. Otherwise, for each cell $\Delta \in L(f)$, we delete f from $\Psi(\mathcal{F}_{i,\Delta})$ recursively. In this case, we do not modify Π_v, so it may contain some of the functions that have been deleted. Following the same analysis as in [6], we can show that the (amortized) deletion time is $O(\psi(n)/n^{1-\varepsilon})$.

Answering a query. Let p be a query point. We again traverse $\Psi(\mathcal{F})$ in a top-down fashion. Suppose we are at a node v. If v is a leaf, we report in $O(\log n + \xi)$ time all ξ functions of $\mathcal{F}^v$ whose graphs lie below p, using the secondary range-searching data structure Π_v (we need to filter out those functions that have already been deleted, but their number is small at a leaf). If v is an internal node then, for each i, we find in $O(\log n)$ time the cell $\Delta_i \in \Xi_i$ that contains p, using the point-location data structure of Lemma 4.2. If Δ_i is defined for every i, we recursively search in $\Psi(\mathcal{F}_{i,\Delta})$, for all $i \leq t$. Otherwise, there is an $i \leq t$ such that p does not lie in any cell of Ξ_i. Let ξ_v be the number of functions in $\mathcal{F}^v$ whose graphs lie below p. Then $\xi_v > m_v/r - m_v/2r = m_v/2r$, because initially Ξ_i was a $(1/r_i)$-cutting of $\mathcal{A}_{\leq m_v/r}(\overline{\mathcal{F}}_i)$ and at most $m_v/2r$ functions of $\mathcal{F}_v$ have been deleted since $\Psi(\mathcal{F}^v)$ was constructed the last time. We query Π_v and report those functions of the query output that have not been deleted so far. If the query outputs consists of ξ'_v functions, then $\xi'_v < \xi_v + m_v/2r = O(\xi_v)$. Hence, the time spent at v is $O(\log n + \xi_v)$. The total query time is thus $O(\log n + \xi)$, where ξ is the total output size. We have thus shown the following.

Theorem 4.3 *Given a family* $\mathbf{F}$ *of* n *bivariate functions satisfying (F1) and (F3) and a subset* $\mathcal{F} \subseteq \mathbf{F}$ *of size* n*, we can preprocess* $\mathcal{F}$ *(in an appropriate model of computation), in time* $O(\psi(n) \cdot n^{\varepsilon})$*, into a data structure of size* $O(\psi(n) \cdot n^{\varepsilon})$ *so that all* ξ *functions whose graphs lie below a query point can be reported in time* $O(\log n + \xi)$*, and a function* $f \in \mathbf{F}$ *can be inserted into or deleted from* $\mathcal{F}$ *in* $O(\psi(n)/n^{1-\varepsilon})$ *(amortized) time. The same dynamic data structure can be used to determine, in* $O(\log n)$ *time, whether a query point lies below the graphs of all the functions in the current set* $\mathcal{F}$.

5 Computing the ($\leq k$)-Level

Next, consider the problem of computing $\mathcal{A}_{\leq k}(\mathcal{F})$, where $\mathcal{F} \subseteq \mathbf{F}$ is a set of n bivariate functions satisfying (F1) and (F3), and k is any integer $< n$. If $\mathcal{F}$ is a set of linear functions in $\mathbb{R}^3$, this can be done by the randomized algorithm of [1] (see also [57, 56]), whose expected time is $O(nk^2 + n \log^3 n)$, but this algorithm does not extend to nonlinear functions. The algorithm that we present below computes all 0-, 1-, and 2-dimensional faces of $\mathcal{A}_{\le k}(\mathcal{F})$, along with their incidence relations. A slight enhancement of the algorithm can store $\mathcal{A}_{\le k}(\mathcal{F})$ into a data structure so that for a query point p, we can determine in $O(\log n)$ time whether $\mu_{\mathcal{F}}(p) \le k$, and, if so, we return the value of $\mu_{\mathcal{F}}(p)$. We construct $\mathcal{A}_{\le k}(\mathcal{F})$ using a divide-and-conquer algorithm, based on our shallow cutting theorem, as follows. (A similar algorithm was developed by Clarkson [23] for computing the ($\le k$)-level in an arrangement of hyperplanes.)

Choose a sufficiently large constant integer parameter r. If $k \ge n/r$, construct the entire arrangement $\mathcal{A}(\mathcal{F})$ and output the first k levels. The time and storage required are $O(n^3 \log n)$ and $O(n^3)$, respectively. (This can be done using one of several known techniques, e.g., for each function $f \in \mathcal{F}$, we compute the intersection curves of f with all the other functions of $\mathcal{F}$ and then compute, by a line-sweep algorithm that takes $O(n^2 \log n)$ time, all the faces of $\mathcal{A}_{\le k}(\mathcal{F})$ that lie on the graph of f.)

Otherwise, the parameter $q = 1 + kr/n$ is at most 2. Using Theorem 3.1, we compute in $O(n)$ time a $(1/r)$-cutting Ξ of size $O(\psi(r))$ for $\mathcal{A}_{\le k}(\mathcal{F})$. For each cell $\Delta \in \Xi$, compute the set $\mathcal{F}_\Delta$ of the functions whose graphs cross Δ, and the set $\mathcal{F}_\Delta^-$ of the functions whose graphs pass fully below Δ. If $|\mathcal{F}_\Delta^-| > k$, then discard Δ. For the remaining cells, we have $|\mathcal{F}_\Delta| \le n/r$ and $k' = |\mathcal{F}_\Delta^-| \le k$. We now construct recursively $\mathcal{A}_{\le k-k'}(\mathcal{F}_\Delta)$, repeat this construction over all cells of Ξ, and glue together the resulting outputs. (The gluing step involves merging those faces which are computed by separate subproblems and which are portions of the same face of $\mathcal{A}(\mathcal{F})$. This can be accomplished in time proportional to the total output size by a rather straightforward procedure, operating on each surface separately, whose details are omitted.)

Let $T(n,k)$ denote the maximum time needed to construct $\mathcal{A}_{\le k}(\mathcal{F})$, for $|\mathcal{F}| = n$. Then we obtain the following recurrence:

$$T(n,k) \le \begin{cases} c_1 \psi(r) \cdot T\left(\dfrac{n}{r}, k\right) + c_2 n & \text{for } k < n/r, \\ c_3 n^3 \log n & \text{for } k \ge n/r, \end{cases}$$

where c_1, c_2, c_3 are appropriate constants. It is easily seen, arguing as in the analysis leading to Theorem 4.1, that the solution of this recurrence is

$$T(n,k) = O(k^3 n^\varepsilon \psi(n/k)),$$

which is close to optimal in the worst case. Hence, we obtain the following result.

Theorem 5.1 *Let* $\mathbf{F}$ *be a family of bivariate functions satisfying (F1) and (F3). Given a set* $\mathcal{F} \subseteq \mathbf{F}$ *of size* n *and an integer parameter* $k \le n$, *we can construct* $\mathcal{A}_{\le k}(\mathcal{F})$ *in time* $O(k^3 n^\varepsilon \psi(n/k))$, *in an appropriate model of computation.*

If we want to store $\mathcal{A}_{\le k}(\mathcal{F})$ into a data structure for answering point-location queries of the form described above, we keep the cuttings obtained during the above recursive construction in a tree-like structure T. This yields a data structure similar to the one described in the previous section.

For a point p, we answer a query as follows. We trace a path in T from the root down, maintaining a global count of the number of functions that are known to pass below p. Initially we set this count to 0. When we visit a node v, we locate p (by brute force) in the cutting Ξ_v associated with v. If no cell of the cutting contains p, we report that p lies above the k-level and stop. Otherwise, let Δ be the cell of Ξ_v containing p. We add $k' = |\mathcal{F}_\Delta^-|$ to the global count, and continue the search at the child of v associated with Δ. When we reach a leaf of T, we explicitly test each of the constant number of functions stored at that leaf whether it passes below p, update the global count accordingly, and output its final value. We thus obtain the following result.

Theorem 5.2 *Given a collection $\mathcal{F}$ of n bivariate functions satisfying (F1)–(F3) and an integer parameter $k < n$, we can preprocess $\mathcal{A}_{\le k}(\mathcal{F})$ into a data structure of size $O(k^3 n^\varepsilon \psi(n/k))$, in time $O(k^3 n^\varepsilon \psi(n/k))$, so that, for any query point p, we can determine in $O(\log n)$ time whether $\mu_\mathcal{F}(p) \le k$, and, if so, we compute the value of $\mu_\mathcal{F}(p)$.*

6 Nearest-Neighbor Searching

Let $S = \{p_1, \ldots, p_n\}$ be a set of n points in the plane, and let $\delta(\cdot,\cdot)$ be a given 'distance function' defined on $\mathbb{R}^2 \times \mathbb{R}^2$. Normally, we regard δ as a metric, or as a convex distance function, but many of the results obtained below apply for more general 'reasonable' functions δ. The function δ is called *reasonable* if the family $\mathbf{F} = \{\delta((x,y),(a_1,a_2)) \mid (a_1,a_2) \in \mathbb{R}^2\}$ satisfies conditions (F1) and (F3) of Section 2. For example, all L_p metrics, for integer $1 \le p \le \infty$, are reasonable. Here we do not assume any metric-like properties of δ, so δ can be fairly arbitrary. We want to store S into a data structure so that points can be inserted into or deleted from S and, for any query point q, we can efficiently compute a *nearest neighbor* of q in S under the function δ (i.e., we want to return a point $p_i \in S$ such that $\delta(q, p_i) = \min_{1 \le j \le n} \delta(q, p_j)$). Let

$$\mathcal{F} = \{f_i(x,y) = \delta((x,y),(\xi_i,\eta_i)) \mid p_i = (\xi_i,\eta_i),\ 1 \le i \le n\}.$$

For a given point q, finding a nearest neighbor of q in S is equivalent to computing a function of $\mathcal{F}$ that attains the lower envelope $E_\mathcal{F}$ of $\mathcal{F}$ at q. Notice that the complexity of the lower envelope $E_\mathcal{F}$ is the same as the complexity of the Voronoi diagram of S under the distance function δ [31], if δ is indeed a metric or a convex distance function.

The nearest-neighbor searching problem, also known as the post-office problem, has been widely studied because of its numerous applications [4, 6, 23, 52], but most of the work to date deals with the case where δ is the L_1, L_2, or L_∞ metric and where S does not change dynamically. If δ is a reasonable metric, then, using the technique of Clarkson [23] and the range searching structure of Chazelle et al. [18], a nearest-neighbor query can be answered in $O(\log^2 n)$ time using $O(n^{3+\varepsilon})$ storage, but this data structure is difficult to dynamize. On the other hand, using the range-searching data structures of Agarwal and Matoušek [5], S can be preprocessed in $O(n \log n)$ time into a linear-size data structure so that a query can be answered in $O(n^{1/2+\varepsilon})$ time. This data structure can be updated in $O(\log^2 n)$ time per insertion or deletion. Since the Agarwal-Matoušek technique will be useful for our applications, we describe it briefly. Let $\Gamma = \{\delta((x,y),(\alpha,\beta)) \le r \mid \alpha,\beta,r \in \mathbb{R}\}$ be the set of all 'disks' under the distance function δ. We preprocess S, in $O(n \log n)$ time, into a linear size data structure for the counting version of Γ-range searching (in which, for any given $\gamma \in \Gamma$, we want to compute $|S \cap \gamma|$), as described in [5]. Using this data structure, for a query point q and a real parameter r, we can determine in $O(n^{1/2+\varepsilon})$ time whether the distance between q and its nearest neighbor in S is less than, greater than, or equal to r. Plugging this procedure into the parametric-searching technique of Megiddo [54] (in the same manner as in [4]), we can answer a nearest-neighbor query, under the distance function δ, in $O(n^{1/2+\varepsilon})$ time. This structure, however, fails to answer a query in $O(\log n)$ time even for the Euclidean metric.

In this section we present a data structure, based on Theorem 4.3, that uses only $O(\psi(n) \cdot n^{\varepsilon})$ space and preprocessing time, answers a query in $O(\log n)$ time, and inserts or deletes a point in $O(\psi(n)/n^{1-\varepsilon})$ time. These bounds are significantly better than the previously mentioned bounds. For example, we can obtain, for the case of L_p metrics, a nearest-neighbor-searching data structure of size $O(n^{1+\varepsilon})$ that can answer a query in $O(\log n)$ time and that can insert/delete a point in $O(n^{\varepsilon})$ time. Compared with the data structure described in [6], we improve the query time significantly at a slight increase in the update time and in the size of the data structure.

As in Section 4, we first describe a static data structure. We preprocess S, or, rather, the associated collection $\mathcal{F}$, into the range-searching data structure $\Pi(\mathcal{F})$ of Theorem 4.1. For a point $q \in \mathbb{R}^2$, we find a function of $\mathcal{F}$ that attains $E_{\mathcal{F}}$ at q, and thus answer a nearest neighbor of q in S, as follows. Let $\vec{\ell}$ be the vertical line in $\mathbb{R}^3$ passing through q, and oriented in the $(+z)$-direction. We trace a path in the tree-structure $T = \Pi(\mathcal{F})$, starting from the root. Suppose we are at a node $v \in T$. Recall that v is associated with a subset $\mathcal{F}^v \subseteq \mathcal{F}$ and a weakly-primitive cell Δ_v. Inductively, assume that $E_{\mathcal{F}}(q) = E_{\mathcal{F}^v}(q)$. If v is a leaf, we explicitly search for a function of $\mathcal{F}^v$ that attains $E_{\mathcal{F}^v}$ at q, report that function and terminate the query. Recall that if v is not a leaf, then each child of v is associated with a cell of a $(1/r)$-cutting Ξ_v of $\mathcal{A}_{\le k}(\mathcal{F}^v)$ (for appropriate parameters r and k). We determine the highest cell Δ of Ξ_v (in the $(+z)$-direction) intersected by $\vec{\ell}$. Since Ξ_v covers $\mathcal{A}_{\le k}(\mathcal{F}^v)$, since Δ is the last cell in Ξ_v intersecting $\vec{\ell}$, and since $\mathcal{F}_{\Delta}$ contains all functions of $\mathcal{F}^v$ whose graphs either intersect Δ or lie below Δ, it easily follows that $E_{\mathcal{F}_{\Delta}}(q) = E_{\mathcal{F}^v}(q) = E_{\mathcal{F}}(q)$. Hence, we can recursively search at the child corresponding to Δ. The overall query time is obviously $O(\log n)$.

If S is allowed to change dynamically, we use the dynamic data structure described in Section 4, with the following modification. For a $(1/r)$-cutting Ξ, we now need a data structure that, for a given query point q in the xy-plane, can determine in $O(\log |\Xi|)$ time the highest cell of Ξ intersected by the vertical line $\vec{\ell}$ passing through q. As in Lemma 4.2, we project the facets of cells in Ξ onto the xy-plane, let Ξ^* be the set of resulting regions in the plane, and compute the arrangement $\mathcal{A}(\Xi^*)$. Since the (interiors of) cells in Ξ are pairwise disjoint, the highest cell of Ξ intersected by a vertical line erected from any point within a face $\phi \in \mathcal{A}(\Gamma)$ is the same, so by storing this cell for each face ϕ, and by preprocessing $\mathcal{A}(\Xi^*)$ for planar point location queries, we can compute in $O(\log |\Xi|)$ time the highest cell of Ξ intersected by a vertical line.

A query is answered as follows: Suppose we are at a node v. If v is a leaf, we compute $E_{\mathcal{F}^v}(q)$, using the procedure just described. Otherwise, for each Ξ_i, we find in $O(\log n)$ time the highest cell Δ intersected by $\vec{\ell}$ by locating q in $\mathcal{A}(\Xi_i^*)$ (as in Lemma 4.2). For each $1 \leq i \leq t$, we recursively compute the function f_i that attains the lower envelope $E_{\mathcal{F}_{i,\Delta}}$ at q. Finally, we return the function that attains the envelope of $\{f_1, \ldots, f_t\}$ at q. Since the procedure visits $O(1)$ nodes and spends $O(\log n)$ time at each node, the overall query time is $O(\log n)$.

Theorem 6.1 *Let S be a set of n points in the plane and let δ be any reasonable distance function, as above. We can preprocess S in time $O(\psi(n)n^\varepsilon)$ into a data structure of size $O(\psi(n)n^\varepsilon)$ so that a nearest-neighbor query in S can be answered in $O(\log n)$ time. Moreover, we can insert and delete points into/from S in $O(\psi(n)/n^{1-\varepsilon})$ time per update operation. Here $\psi(n)$ is an upper bound, satisfying (F3), for the maximum complexity of the envelope $E_{\mathcal{F}}$, as defined above (if δ is a metric or a convex distance function, then $\psi(n)$ is also (an upper bound for) the maximum complexity of the Voronoi diagram of a set of n points in the plane, under the distance function δ).*

Remark 6.2 For a point q, finding a farthest neighbor of q in S is equivalent to finding a function of $\mathcal{F}$ that attains the upper envelope of $\mathcal{F}$ at q. By reversing the direction of the z-axis, we can use a similar data structure, with similar performance bounds, for answering farthest-neighbor queries.

Reporting the κ nearest (farthest) neighbors. The above query procedures can be modified so that for a query point q and an integer $\kappa \leq n$, we can compute the κ nearest (or farthest) neighbors of q in S in time $O(\log n + \kappa)$. Consider the static data structure, for instance. We again follow a path of the tree, starting from the root. Suppose we are at a node v. If v is a leaf or $|\mathcal{F}^v| \leq 3\kappa r$, we explicitly check all functions of $\mathcal{F}^v$ to determine the κ nearest neighbors of q in S, and terminate the query. Otherwise, we find the highest cell of Ξ_v intersected by the corresponding vertical line $\vec{\ell}$, and recursively visit that child. For the dynamic version, the query procedure is somewhat more involved, but a query can still be answered in $O(\log n + \kappa)$ time. Hence, we can conclude:

Theorem 6.3 *Let S be a set of n points in the plane and let δ be any reasonable distance function, as above. We can preprocess S in time $O(\psi(n)n^\varepsilon)$ into a data structure of size*

$O(\psi(n)n^{\varepsilon})$ so that for a given point p and an integer $\kappa \leq n$, the κ nearest (or farthest) neighbors of p can be computed in $O(\log n + \kappa)$ time. Moreover, a point can be inserted into or deleted from S in $O(\psi(n)/n^{1-\varepsilon})$ time.

The case of L_p-metrics and weighted Euclidean metrics. If δ is an L_p-metric, for any integer $1 \leq p \leq \infty$, then $\psi(n) = O(n)$ [46]. Similarly, if δ is an *(additive) weighted Euclidean* metric, i.e., each point $p_i \in S$ has a weight w_i and $\delta(q, p_i) = d(q, p_i) + w_i$, where $d(\cdot, \cdot)$ is the Euclidean distance, then also $\psi(n) = O(n)$ [47]. Hence, we obtain the following result.

Corollary 6.4 *Let S be a set of n points in the plane, and let δ be any L_p-metric or any weighted Euclidean metric. We can preprocess S in $O(n^{1+\varepsilon})$ time into a data structure of size $O(n^{1+\varepsilon})$ so that points can be inserted into or deleted from S in time $O(n^{\varepsilon})$ per update, and a nearest-neighbor query can be answered in $O(\log n)$ time. We can also construct a data structure with the same performance for the case when S is a set of n disjoint segments in the plane and δ is the Euclidean metric.*

The general case: Space/query-time tradeoff. If the bound $\psi(n)$ is rather large, say, $O(n^{2+\varepsilon})$, we can obtain a space/query-time tradeoff by combining Theorem 6.1 with the linear-size data structure of [5] for nearest-neighbor searching, mentioned at the beginning of this section, in a rather standard manner [22]. For example, if $\psi(n) = O(n^{2+\varepsilon})$ then, for any parameter $n \leq m \leq n^2$, we can store S into a data structure of size $O(m^{1+\varepsilon})$ so that a nearest-neighbor query can be answered in time $O(n^{1+\varepsilon}/\sqrt{m})$. Insertions and deletions can be performed in $O(m^{1+\varepsilon}/n)$ time, per update. Hence, we can conclude the following.

Theorem 6.5 *Let S be a set of n points in the plane, $n \leq m \leq n^2$ a real parameter, and δ a reasonable distance function, as above. We can preprocess S in time $O(m^{1+\varepsilon})$ into a data structure of size $O(m^{1+\varepsilon})$ so that a nearest-neighbor query can be answered in $O(n^{1+\varepsilon}/\sqrt{m})$ time. Moreover, we can insert and delete points in $O(m^{1+\varepsilon}/n)$ time per update.*

Remark 6.6 If the number of queries and the number of updates are both $O(n)$, then an optimal choice for m is $n^{4/3}$, for which the entire sequence of operations can be performed in $O(n^{4/3+\varepsilon})$ time.

The dynamic bichromatic closest-pair problem. Next, we consider the following *dynamic bichromatic closest-pair* problem: Let R and B be two sets of points in the plane, of a total of at most n points. We wish to update R and B dynamically and to return the closest pair between R and B after each update operation.[4] We use the following result of Eppstein [35]:

[4] A *closest pair* between R and B under a distance function δ is a pair of points $p \in R, q \in B$ such that $\delta(p,q) = \min_{p' \in R, q' \in B} \delta(p', q')$.

Lemma 6.7 *Let R and B be two sets of points in the plane, of a total of at most n points. Let δ be a distance function for which we can store a set of n points into a data structure that supports insertions, deletions, and nearest neighbor queries in $O(T(n))$ time per operation. Then a closest pair between R and B can be maintained in $O(T(n)\log n)$ time per insertion and in $O(T(n)\log^2 n)$ time per deletion.*

If δ is an L_p metric or any weighted Euclidean metric, then, by Corollary 6.4, $T(n) = O(n^\varepsilon)$. On the other hand, if δ is any reasonable metric, then, by choosing $m = n^{4/3}$ in Theorem 6.5, $T(n) = O(n^{1/3+\varepsilon})$. Plugging these values into the above lemma, we obtain the following result.

Theorem 6.8 *Let R and B be two sets of points in the plane with a total of n points. We can store $R \cup B$ in a dynamic data structure of size $O(n^{1+\varepsilon})$ that maintains a closest pair in $R \times B$, under any L_p-metric or any weighted Euclidean metric, in $O(n^\varepsilon)$ time per insertion or deletion. If we use an arbitrary, reasonable distance function, then the storage increases to $O(n^{4/3+\varepsilon})$ and the update time to $O(n^{1/3+\varepsilon})$.*

Using this theorem and extending the analysis of Agarwal et al. [2], as done in [35], we can obtain an efficient procedure for maintaining a minimum spanning tree (MST) T of a set of points under any L_p metric, as follows. For an angle $\alpha \leq \pi/2$ and a unit vector d, let $Cone(d, \alpha) = \{\mathbf{x} \mid \angle(\mathbf{x}, d) \leq \alpha\}$. We call a pair of point sets P, Q *α-separated* if there exist a point z and a direction d such that $P \subseteq z + Cone(-d, \alpha)$ and $Q \subseteq z + Cone(d, \alpha)$. Let α_0 be a sufficiently small angle, and let

$$\mathcal{C} = \{(P_1, Q_1), \ldots, (P_u, Q_u)\}$$

be a family of α_0-separated pairs such that, for every pair $p, q \in S$, there is an i with $p \in P_i$ and $q \in Q_i$; we will refer to $\mathcal{C}$ as an *α_0-cover* of S. Consider the graph G whose vertices are the points of S and whose edges are pairs of points (p_i, q_i), $1 \leq i \leq u$, such that (p_i, q_i) is a closest pair between P_i and Q_i. The weight of an edge (p_i, q_i) is $\delta(p_i, q_i)$. Following the same argument used by Agarwal et al. [2, Lemmas 1–4], but extending their Lemma 3 to the case of an L_p-metric, one can show that G contains a minimum spanning tree of S as a subgraph.

Using a standard divide-and-conquer approach, an α_0-cover $\mathcal{C}$ of S, with the property that each point of S is contained in $O(\log^2 n)$ pairs, can be constructed in $O(n \log^2 n)$ time; see e.g., [2]. Moreover, $\mathcal{C}$ can be maintained in $O(\log^2 n)$ time per insertion or deletion. Insertion or deletion of a point changes $O(\log^2 n)$ edges of the graph G, and Theorem 6.8 implies that the new edges can be found in $O(n^\varepsilon \log^2 n)$ time. The problem of maintaining am MST of S now reduces to maintaining the family $\mathcal{C}$, a closest pair for each pair $(P_i, Q_i) \in \mathcal{C}$, and an MST of the resulting graph G. Eppstein et al. [36] have shown that an MST of a graph with n vertices can be updated in $O(\sqrt{n})$ time for each insertion or deletion of an edge into/from the graph. Hence we obtain:

Theorem 6.9 *A minimum spanning tree of a set of n points in the plane, under any L_p metric, can be maintained in $O(\sqrt{n}\log^2 n)$ time for each insertion or deletion operation, using $O(n^{1+\varepsilon})$ storage.*

7 Minimum-Weight Bipartite Euclidean Matching

Let $P = \{p_1, \ldots, p_n\}$ and $Q = \{q_1, \ldots, q_n\}$ be two sets of points in the plane. We wish to compute a *minimum-weight bipartite Euclidean matching* M of P and Q. A matching of P and Q is a set of n pairs $(p, q) \in P \times Q$ such that each point of $P \cup Q$ appears in exactly one pair. The weight of a pair is the Euclidean distance between its points, and the weight of a matching is the sum of the weights of its pairs. The standard *Hungarian method* [44, 45] yields an $O(n^3)$-time algorithm for computing M. Exploiting the fact that the weights are Euclidean distances, Vaidya obtained an $O(n^{2.5}\log n)$-time algorithm for computing M [67]. A number of efficient algorithms have been developed for computing a minimum-weight bipartite Euclidean matching when P and Q have some special structure [10, 17, 48], but no progress has been made when P and Q are arbitrary sets of points in the plane. Recently, Alon and Itai [33] have proposed an $O(n^{1.5+\varepsilon})$-time algorithm for computing a bottleneck Euclidean bipartite matching, in which one wishes to minimize the maximum weight of an edge in the matching.

In this section, we show that Vaidya's algorithm, for arbitrary sets A, B, can be modified so that its running time improves to $O(n^{2+\varepsilon})$. One can equally consider the nonbipartite version of the problem, where we are given just one set P of $2n$ points in the plane, and wish to compute a minimum-weight Euclidean matching in P, that is, a partition of P into n pairs of points, so that the sum of the intra-pair Euclidean distances is minimized. Vaidya has also given in [67] an $O(n^{2.5}\log n)$-time algorithm for the nonbipartite case, which, so far, we do not know how to improve.

For the sake of completeness, we first give a brief sketch of Vaidya's algorithm for the bipartite case, and then show how it can be improved using the new machinery of this paper.

The bipartite matching problem can be formulated as a linear program:

$$\begin{array}{lll} \min & \sum_{i,j} d(p_i, q_j) x_{ij}\,, \quad \text{subject to} & \\ & \sum_{j=1}^{n} x_{ij} = 1, & i = 1, \ldots, n, \\ & \sum_{i=1}^{n} x_{ij} = 1, & j = 1, \ldots, n, \\ & x_{ij} \geq 0, & i, j = 1, \ldots, n, \end{array}$$

where (p_i, q_j) is an edge of M if and only if $x_{ij} = 1$. The dual linear program is

$$\begin{aligned} \max \quad & \sum_i \alpha_i + \sum_j \beta_j\,, \quad \text{subject to} \\ & \alpha_i + \beta_j \le d(p_i, q_j), \qquad\qquad i, j = 1, \ldots, n, \end{aligned}$$

where α_i (resp. β_j) is the dual variable associated with the point p_i (resp. q_j).

The Hungarian method computes a matching in n phases, each of which augments the matching by one edge and updates the dual variables. Let X be the current matching computed so far by the algorithm. Initially, we set $X = \emptyset$, $\alpha_i = 0$, and $\beta_j = \min_i\, d(p_i, q_j)$ for all i, j. An edge (p_i, q_j) is called *admissible* if $d(p_i, q_j) = \alpha_i + \beta_j$. A vertex of $P \cup Q$ is called *exposed* if it is not incident to any edge of X. An *alternating path* is one that alternately traverses edges of X, and edges not in X, starting with an edge not in X. An alternating path between two exposed vertices is called an *augmenting path*.

During each phase, we search for an augmenting path consisting only of admissible edges, as follows. For each exposed point $q \in Q$, we grow (in an implicit manner) an 'alternating tree' whose paths are alternating paths starting at q. More precisely, each point of $P \cup Q$ in an alternating tree is reachable from its root by an alternating path that consists only of admissible edges. For a point w of P (resp. Q), the path leading to w ends at an edge not in X (resp. in X). Let S (resp. T) denote the set of points of Q (resp. P) that lie in any alternating tree. In the beginning of each phase, S is the set of exposed vertices of Q and $T = \emptyset$. Let

$$\delta = \min_{p_i \in P - T, q_j \in S} \{d(p_i, q_j) - \alpha_i - \beta_j\}.$$

At each step, the algorithm takes one of the following actions, depending on whether $\delta = 0$ or $\delta > 0$:

Case 1: $\delta = 0$. Let (p_i, q_j), for $p_i \in P - T$ and $q_j \in S$, be an admissible edge ($\delta = 0$ implies that such an edge must exist). If p_i is an exposed vertex, an augmenting path has been found, so the algorithm moves to the next phase (see below for more details). Otherwise, let q_k be the vertex such that $(p_i, q_k) \in X$. The algorithm adds the edges (p_i, q_j) and (p_i, q_k) to all the alternating trees that contain the vertex q_j. It also adds the point p_i to T and the point q_k to S.

Case 2: $\delta > 0$. The algorithm updates the dual variables, as follows. For each vertex $p_i \in T$, it sets $\alpha_i = \alpha_i - \delta$ and, for each $q_j \in S$, it sets $\beta_j = \beta_j + \delta$.

The algorithm repeats these steps until it reaches an exposed vertex of P, thereby obtaining an augmenting path. If an augmenting path Π is found, we delete the edge of $\Pi \cap X$ from the current matching X, add the other edges of Π to X (thereby increasing the size of the current matching by 1), and move to the next phase. This completes the description of the algorithm. Further details of the algorithm and the proof of its correctness can be found in [45, 67].

For arbitrary graphs, each step can be implemented in $O(n)$ time. Since there are at most n phases and each phase consists of $O(n)$ steps, the total running time of the above procedure is $O(n^3)$. Vaidya suggested the following approach to expedite the running time of each step. Maintain a variable Δ and associate a weight $w(\cdot)$ with each point in $P \cup Q$. In the beginning of each phase, $\Delta = 0$ and $w(p_i) = \alpha_i$, $w(q_j) = \beta_j$ for each $1 \le i, j \le n$. During each step, the weights and Δ are updated, but the values of the dual variables remain unchanged. This is done as follows. If Case 1 occurs, then we set $w(p_i) = \alpha_i + \Delta$ and $w(q_k) = \beta_k - \Delta$, and do not change the value of Δ. If Case 2 occurs, then we set $\Delta = \Delta + \delta$, and do not change the weights. Notice that, for each $q_j \in S$, the current value of β_j is equal to $w(q_j) + \Delta$, and, similarly, for each $p_i \in T$, the current value of α_i is equal to $w(p_i) - \Delta$. Also, the current value of the dual variables for other points is equal to their values at the beginning of the phase. At the end of each phase, the value of the dual variables can be computed from Δ and from the weights of the corresponding points. The weight of each point changes only once during a phase, namely, when it is added either to S or to T. Moreover, at any time during a phase,

$$\delta = \min_{p_i \in P-T, q_j \in S} \{d(p_i, q_j) - w(p_i) - w(q_j)\} - \Delta \, .$$

Hence, δ can be computed during each step by maintaining the weighted closest pair between S and $P - T$. Since each step requires at most two update operations (inserting a point into S and deleting a point from $P - T$), each step can be performed in $O(n^\varepsilon)$ time, by the algorithm provided in Theorem 6.8. (The current setup is slightly different than that of Section 6, because here the weighted distance between two points subtracts weights associated with both points. Nevertheless, the analysis of Section 3 leading to Theorem 4.3, and the way in which this theorem is used in Section 6 still apply.) If the distance between two points is measured by an arbitrary reasonable distance function, each step of the above algorithm will require $O(n^{1/3+\varepsilon})$ time, with an appropriately increased storage (the distance function continues to be reasonable, when weights are added to the points, as above). We can thus conclude:

Theorem 7.1 *Given two sets P, Q, each consisting of n points in the plane, a minimum-weight bipartite Euclidean matching between P and Q can be computed in $O(n^{2+\varepsilon})$ time, using $O(n^{1+\varepsilon})$ space. If we use an arbitrary, reasonable distance function for computing distances between points of P and Q, a minimum weight bipartite matching between P and Q can be computed in time $O(n^{7/3+\varepsilon})$, using $O(n^{4/3+\varepsilon})$ space.*

The above algorithm can be extended to obtain faster algorithms for several other problems related to the bipartite matching. For example, consider the following *Euclidean transportation* problem, which is a generalization of the minimum-weight bipartite Euclidean matching problem: Let $P = \{p_1, \ldots, p_u\}$ bet a set of u 'supply' points in the plane, each with a supply value a_i, and let $Q = \{q_1, \ldots, q_v\}$ be a set of v 'demand' points in the plane, each with a demand value b_j, such that $\sum_{i=1}^{u} a_i = \sum_{j=1}^{v} b_j$. Assume that the cost of transporting unit commodity from p_i to q_j is $d(p_i, q_j)$. We wish to find out how to satisfy all the demands at the minimum cost. Atkinson and Vaidya [13] gave an $O((u+v)^{2.5}\log(u+v)\log N)$-time algorithm, where $N = \max\{\max_i a_i, \max_j b_j\}$. Plugging Theorem 6.8 into their algorithm, we can improve the running time to $O((u+v)^{2+\varepsilon}\log N)$. See [14, 45, 67] for some other applications.

8 Maintaining the Intersection of Congruent Balls in Three Dimensions

The next application of our techniques is an efficient algorithm for dynamically maintaining the intersection of congruent balls in 3-space, under insertions and deletions of balls. With no loss of generality, we assume that the balls have radius 1. For a point $p \in \mathbb{R}^3$, let $B(p)$ denote the ball of radius 1 centered at p. Let S be a set of n points in $\mathbb{R}^3$. Let $\mathcal{B} = \mathcal{B}(S) = \{B(p) \mid p \in S\}$, and let $K(S) = \bigcap_{p\in S} B(p)$ be the common intersection of $\mathcal{B}$. We wish to maintain $K(S)$ as we update S dynamically by inserting and deleting points. Although the complexity of $K(S)$ is $O(n)$ [37], a sequence of m updates in S may cause $\Omega(mn + m^2)$ changes in the structure of $K(S)$, even if we perform only insertions. Consequently, we cannot hope to maintain $K(S)$ explicitly, if we seek fast update time. Instead, we store $K(S)$ into a data structure so that a number of different types of queries can be answered efficiently. The simplest query is whether a query point p lies in $K(S)$. A stronger type of query is to determine efficiently, after each update, whether $K(S)$ is empty. In this section, we present a data structure that answers these two queries efficiently.

Let $B^+(p)$ (resp. $B^-(p)$) denote the region consisting of all points that lie in or above (resp. in or below) $B(p)$. Let $K^+(S) = \bigcap_{p\in S} B^+(p)$ and $K^-(S) = \bigcap_{p\in S} B^-(p)$. If we regard the boundary of each region $B^+(p)$, for $p \in S$, as the graph of a partially-defined bivariate function, then (the nonvertical portion of) $\partial K^+(S)$ is the graph of the upper envelope of these functions. We denote this upper envelope by $z = F^+(\mathbf{x})$. The same argument as in [37] implies that $K^+(S)$ also has linear complexity. In fact, the surfaces $\partial B^+(p)$ behave similar to a family of pseudoplanes—any pair of them intersect in a single connected unbounded curve (or does not intersect at all), and, assuming general position, any triple of them intersect at a single point (or does not intersect at all). Similar and symmetric properties hold for the region $K^-(S)$, and we use the notation $z = F^-(\mathbf{x})$ to denote the graph of its (nonvertical) boundary.

A query point $p = (p_x, p_y, p_z)$ lies in $K(S)$ if and only if $F^+(p_x, p_y) \le p_z \le F^-(p_x, p_y)$. Therefore, the problem of determining whether $p \in K(S)$ reduces to evaluating $F^+(p_x, p_y)$ and $F^-(p_x, p_y)$. By Theorem 6.1, we can store $\{B^+(p) \mid p \in S\}$ and $\{B^-(p) \mid p \in S\}$ into two data structures, each of size $O(n^{1+\varepsilon})$ so that a point can be inserted into or deleted from S in $O(n^\varepsilon)$ time, and so that for a query point $p = (p_x, p_y, p_z)$, the functions $F^+(p_x, p_y)$ and $F^-(p_x, p_y)$ can be evaluated in $O(\log n)$ time.[5]

[5]This is an example where the given bivariate functions are only partially defined. As argued at the

The same data structure can be used to determine whether $K(S)$ is empty, although this is a considerably more involved operation. Observe that the boundaries of the regions $K^+(S)$ and $K^-(S)$ are both (weakly) xy-monotone—one of them is a convex surface and the other is concave. These properties can be exploited to obtain a fast procedure for detecting (after each update) whether $K^+(S)$ and $K^-(S)$ intersect, by applying a variant of the parametric searching technique [54], as proposed by Toledo [66]. For the sake of completeness, we include here a brief overview of this technique.

We wish to find a point $\mathbf{x}^* \in \mathbb{R}^2$ such that $F^+(\mathbf{x}^*) \leq F^-(\mathbf{x}^*)$; this point will serve as a witness to the fact that $K^+(S)$ and $K^-(S)$ intersect. In fact, we will seek a point $\mathbf{x}^*$ at which

$$F^\Delta(\mathbf{x}^*) \stackrel{\text{def}}{=} F^+(\mathbf{x}^*) - F^-(\mathbf{x}^*)$$

is minimized; the minimum is negative if and only if the regions $K^+(S)$ and $K^-(S)$ intersect.

We first consider the 1-dimensional problem where, for a given line $\ell : x = c$, we wish to find a point $\mathbf{y}^* = (c, y^*) \in \ell$ that minimizes F^Δ. We execute the query procedure described in Section 6 to evaluate F^Δ in a generic manner, without knowing the value $\mathbf{y}^*$. This procedure makes $O(\log n)$ comparisons, each of which involves computing the sign of a univariate, constant-degree polynomial $g(y)$ in the y-coordinate y^* of $\mathbf{y}^*$. We compute the roots $y_1, \ldots, y_p$ of g, and evaluate $F^\Delta(c, y_i)$ for $1 \leq i \leq p$. Set $y_0 = -\infty$ and $y_{p+1} = +\infty$. Since F^Δ is convex, by examining the local behavior of $F^\Delta(c, \cdot)$ at each y_i, we can determine whether $\mathbf{y}^* = (c, y_i)$ for some $0 \leq i \leq p+1$, and if not, then we can determine the open interval (y_i, y_{i+1}) that contains y^*. We can thus compute the sign of g at $\mathbf{y}^*$. Proceeding in this manner, we can compute the value of $\mathbf{y}^*$. Since F^Δ is convex over the entire xy-plane, the above procedure can be extended to determine whether the global minimum $\mathbf{x}^*$ of F^Δ lies to the left of ℓ, to the right of ℓ, or on ℓ, simply by examining the local behavior of F^Δ at a neighborhood of $\mathbf{y}^*$. (In the third case, the procedure can return the value of $\mathbf{x}^* = \mathbf{y}^*$ and terminate.) The total running time of this procedure is $O(\log^2 n)$, because we spend $O(\log n)$ time at each comparison, and the algorithm makes $O(\log n)$ comparisons.

Using the above procedure as a subroutine, we can compute $\mathbf{x}^*$ as follows. We now run the query procedure of Theorem 6.1 in a generic manner at $\mathbf{x}^*$. Each comparison now involves computing the sign of a bivariate, constant-degree polynomial $\pi(\mathbf{x})$. We compute the Collins' *cylindrical algebraic decomposition* [12, 26, 59] $\hat{\pi}$ of $\mathbb{R}^2$ into $O(1)$ cells, so that the sign of π is invariant within each cell of $\hat{\pi}$; see Figure 5 (ii). Our aim is to determine the cell $\tau \in \hat{\pi}$ that contains $\mathbf{x}^*$, thereby determining the sign of π at $\mathbf{x}^*$. The cells of $\hat{\pi}$ are delimited by $O(1)$ y-vertical lines. For each vertical line ℓ, we run the above 1-dimensional procedure to determine which side of ℓ contains $\mathbf{x}^*$. If any of these subroutine calls returns $\mathbf{x}^*$, we are done. Otherwise, we know the vertical strip σ that contains $\mathbf{x}^*$.

We still have to search through the cells of $\hat{\pi}$ within σ, stacked one above the other in the y-direction, to determine which of them contains $\mathbf{x}^*$. We note that the number of roots of π along any vertical line $\ell : x = x_0$ within σ is the same, and that each root varies

beginning of Section 2, the preceding results hold in this case too.

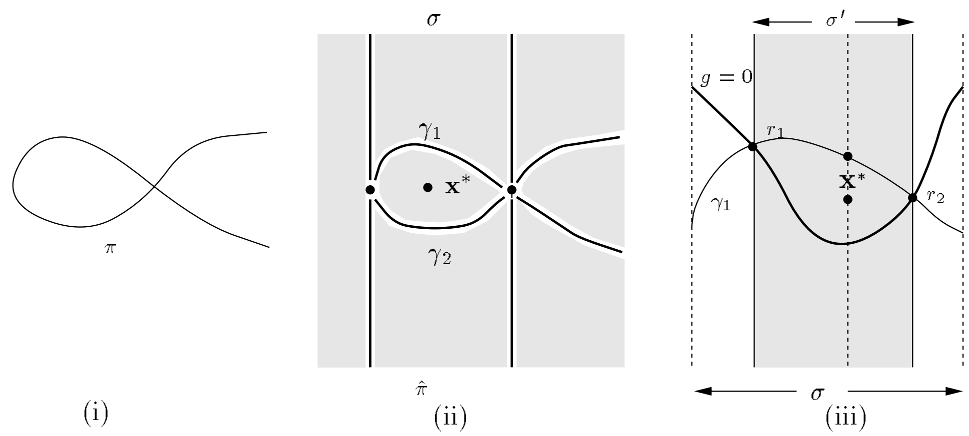

Figure 5: (i) roots of π; (ii) $\hat{\pi}$, the cylindrical algebraic decomposition of π; (iii) the curves $g = 0$ and γ_1

continuously with x_0. That is, the roots of π in σ constitute a collection of x-monotone arcs $\gamma_1, \ldots, \gamma_t$ whose endpoints lie on the boundaries of σ (see Figure 5 (ii)). It suffices to determine, for each γ_i, whether $\mathbf{x}^*$ lies below, above, or on γ_i. This is accomplished by executing the 1-dimensional algorithm generically, on the line $\ell : x = x^*$ that contains $\mathbf{x}^*$. In other words, we run the query procedure of Theorem 6.1 at the generic point $\mathbf{x}_i = \gamma_i \cap \ell$. This time, performing a comparison involves computing the sign of a bivariate, constant-degree polynomial g at $\mathbf{x}_i$ (we prefer to treat g as a bivariate polynomial, although we could have eliminated one variable, by restricting $\mathbf{x}_i$ to lie on γ_i). We compute the roots $r_1, \ldots, r_u$ of g that lie on γ_i (see Figure 5 (iii)), and set r_0 and r_{u+1} to be the left and right endpoints of γ_i, respectively. As above, we compute the index j such that $\mathbf{x}^*$ lies in the vertical strip σ' bounded between r_j and r_{j+1}. Notice that the sign of g is the same for all points on γ_i within the strip σ', so we can now compute the sign of g at $\mathbf{x}_i$. After completing the execution of the 1-dimensional procedure, we know whether $\mathbf{x}^*$ lies above, below, or on γ_i. Repeating these procedure for all γ_i's, we know the cell of $\hat{\pi}$ that contains $\mathbf{x}^*$. The total time spent in computing the sign of $\pi(\mathbf{x}^*)$ is $O(\log^3 n)$, because the 1-dimensional algorithm is composed of $O(\log^2 n)$ steps and the execution of each step requires $O(\log n)$ time. Since the 2-dimensional algorithm also consists of $O(\log n)$ steps, the overall query time is $O(\log^4 n)$. We thus conclude:

Theorem 8.1 *The intersection of a set of congruent balls in 3-space can be maintained dynamically, by a data structure of size $O(n^{1+\varepsilon})$ so that each insertion or deletion of a ball takes $O(n^\varepsilon)$ time, and the following queries can be answered: (a) For any query point p, we can determine in $O(\log n)$ time whether p lies in the current intersection, and (b) after performing each update, we can determine in $O(\log^4 n)$ time whether the current intersection* *is nonempty.*

Remark 8.2 (i) The same data structure can be used to answer other queries, such as determining whether a line intersects $K(S)$, computing the highest (or the lowest) vertex of $K(S)$, etc.

(ii) The same technique can also be used to maintain the intersection of other simply-shaped, convex objects in $\mathbb{R}^3$. The technique will be most effective in cases where the complexity of such an intersection can be shown to be small.

(iii) The problem studied in this section arises in the three-dimensional *2-center* problem, where we are given a set S of n points in $\mathbb{R}^3$, and wish to cover them by the union of two congruent balls whose radius is as small as possible. Extending to three dimensions the standard approach to this problem [29], we face a main subproblem in which we need to maintain dynamically the intersection of a set of congruent balls, and to determine after each update whether this intersection is nonempty. The main difference, though, is that in the 2-center problem the sequence of updates is known in advance, which makes the problem easier to solve.

9 Smallest Stabbing Disk

Let $\mathbf{C}$ be a (possibly infinite) family of simply-shaped, compact, strictly-convex sets (called *objects*) in the plane. By 'simply-shaped' we mean that each object is described by a Boolean combination of a constant number of polynomial equalities and inequalities of constant maximum degree. Let $\mathcal{C} = \{c_1, \ldots, c_n\}$ be a finite subset of $\mathbf{C}$. We wish to update $\mathcal{C}$ dynamically, by inserting objects $c \in \mathbf{C}$ into $\mathcal{C}$ or by deleting such objects from $\mathcal{C}$, and maintain a smallest disk, or, more generally, a smallest homothetic copy of some given simply-shaped, compact, convex set P, that intersects all sets of $\mathcal{C}$. This study extends recent work by Agarwal and Matoušek [6] who have obtained an algorithm that takes $O(n^\varepsilon)$ time per update operation for the case where $\mathcal{C}$ is a set of points and P is a disk.

9.1 Farthest-neighbor Voronoi diagrams

The set P induces a *convex distance function* defined by

$$d_P(x, y) = \min\{\lambda \mid y \in x + \lambda P\};$$

(we assume here that P contains the origin in its interior). The function d_P is a metric if and only if P is centrally symmetric with respect to the origin. We define the farthest-neighbor Voronoi diagram $Vor_P(\mathcal{C})$ of $\mathcal{C}$, under the distance function d_P, in a standard manner (see [43] for details). In what follows we assume that P is strictly convex; the results can be extended to more general sets with some extra care. We first prove the following theorem, which is of independent interest (we are not aware of any previous proof of this result).

Theorem 9.1 *Let $\mathcal{C}$ be a set of n (possibly intersecting) simply-shaped, compact, convex objects in the plane. Then the complexity of the farthest-neighbor Voronoi diagram of $\mathcal{C}$, under a convex distance function d_P induced by any simply-shaped, compact, strictly-convex set P, is $O(\lambda_s(n))$. Here s is a constant depending on the shape of P and of the objects in $\mathcal{C}$. If $\mathcal{C}$ is a set of n line segments and d_P is the Euclidean distance function (i.e., P is a disk), the complexity of the farthest-neighbor Voronoi diagram is $O(n)$.*

Proof: Let R be one of the Voronoi cells of $Vor_P(\mathcal{C})$, and let $c \in \mathcal{C}$ be the farthest neighbor of all points of R. We show that R has the following 'anti-star-shape' property: Let $x \in R$, and let $q \in c$ be the nearest point to x (i.e., $d_P(x,q) = d_P(x,c)$); the convexity of c and the strict convexity of P imply that q is unique. Let ρ be the ray emanating from q towards x, and let y be any point on ρ past x (i.e., x lies in the segment qy). We claim that c is the farthest neighbor of y; see Figure 6. Indeed, suppose to the contrary that the farthest neighbor of y in $\mathcal{C}$ is $c' \neq c$ (so that $d_P(y,c') > d_P(y,c)$), and let $r \in c'$ be the nearest point to x. Let ℓ be the (unique) line passing through q and tangent to $x + d_P(x,c)P$ there. The convexity of c implies that c is contained in the (closed) halfplane bounded by ℓ and not containing x. This in turn is easily seen to imply that $d_P(y,c) = d_P(y,q)$. Now, by the triangle inequality, we have

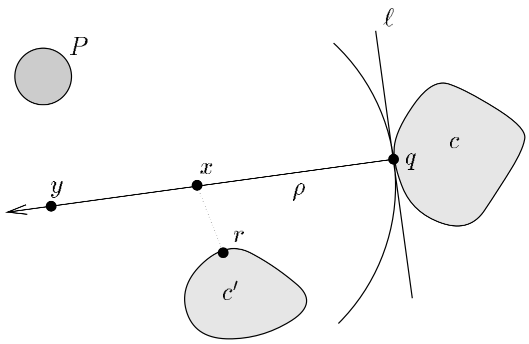

Figure 6: Anti-star-shape property of Voronoi cells

$$\begin{aligned} d_P(y,c') &\leq d_P(y,r) \\ &\leq d_P(y,x) + d_P(x,r) \\ &= d_P(y,x) + d_P(x,c') \\ &\leq d_P(y,x) + d_P(x,c) \\ &= d_P(y,x) + d_P(x,q) \\ &= d_P(y,q) \\ &= d_P(y,c), \end{aligned}$$

a contradiction that establishes the claim.

This implies that R is unbounded, and so $Vor_P(\mathcal{C})$ is an outerplanar map. It is easy to verify that every vertex of $Vor_P(\mathcal{C})$ has degree at least three. Hence, by Euler's formula for planar graphs, the complexity of $Vor_P(\mathcal{C})$ is proportional to the number of faces in it. We bound the number of faces in $Vor_P(\mathcal{C})$, as follows. By a compactness argument, there exists a sufficiently large r such that the circle σ_r of radius r about the origin cuts the cells of $Vor_P(\mathcal{C})$ in the same sequence as does the circle at infinity. For each $c \in \mathcal{C}$, let $f_c(\theta) = d_P((r,\theta), c)$, and let $F(\theta) = \max_{c \in \mathcal{C}} f_c(\theta)$. Clearly, the number of edges of $Vor_P(\mathcal{C})$ crossed by σ_r, and thus the number of faces in $Vor_P(\mathcal{C})$, is equal to the number of breakpoints in the upper envelope F. Since we have assumed P and each set $c \in \mathcal{C}$ to be simply shaped, it follows that, for each pair $c, c' \in \mathcal{C}$, the functions f_c and $f_{c'}$ have at most some constant number, s, of intersection points. Hence, the number of breakpoints of F is, by standard Davenport-Schinzel theory [40, 63], at most $\lambda_s(n)$. This establishes the first part of the claim.

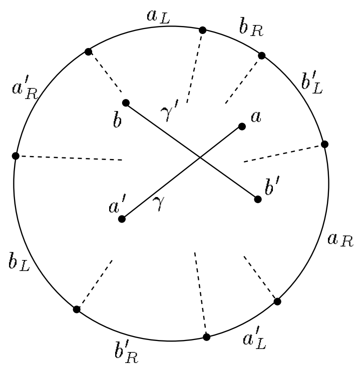

Figure 7: Partition of the circle σ_r into maximal arcs.

For the case of line segments and Euclidean distance, the bound improves to $O(n)$. To see this, let $\mathcal{C}$ be a set of n line segments in the plane. Traverse the circle σ_r, as defined above, and decompose it into maximal arcs so that the following conditions are satisfied for each arc ζ.

(i) The farthest segment from all points $q \in \zeta$ is the same segment e.

(ii) The nearest endpoint of e from q is fixed.

(iii) ζ lies fully in one of the halfplanes containing e.

We label each arc ζ by the endpoint of the corresponding segment e, but we use a different label for arcs that lie on different 'sides' of e (as in (iii) above); see Figure 7. We call an arc ζ, labeled by endpoint a, a *left arc* if it lies on the left side of the segment incident to a, when this segment is directed towards a; otherwise, ζ is a *right arc*. Let S_L (resp. S_R, S) denote the cyclic sequence consisting of the labels of all left (resp. right, all) arcs in the clockwise order along σ_r. Note that S is obtained by merging S_L and S_R, and it does not contain any pair of equal adjacent elements (S_L and S_R may contain such pairs). Each of S_L and S_R is composed of at most $2n$ symbols. We claim that S_L cannot contain an alternating subcycle of the form $\langle a \cdots b \cdots a \cdots b\rangle$, where a and b are two segment endpoints.

In order to prove this claim, it suffices to rule out the existence of such a subcycle in the partition of σ_r induced by the one or two segments incident to a and b. First, such a subcycle is impossible if a and b are endpoints of the same segment e, because all arcs labeled by a are separated from all arcs labeled by b by the perpendicular bisector of e.

Suppose then that a and b are endpoints of two distinct respective segments e_1, e_2. Let a' be the other endpoint of e_1, and let b' be the other endpoint of e_2. Since r is sufficiently large, the perpendicular bisector of any two endpoints of segments in $\mathcal{C}$ partitions σ_r into two arcs, each of which spans an angle more than, say, $3\pi/4$. Both appearances of a (resp. of b) in the assumed subcycle occur within an arc γ_1 (resp. γ_2), bounded by the perpendicular bisector of a and b and by the line containing e_1 (resp. e_2). See Figure 8. An easy calculation shows that the angles spanned by γ_1, γ_2 are less than $3\pi/4$ (assuming that r is chosen sufficiently large), and by definition, a is the nearest endpoint of e_1 within γ_1 and b is the nearest endpoint of e_2 within γ_2. It follows that γ_1 and γ_2 overlap, and that their intersection σ^* contains (at least) the two middle elements of the subcycle. (Since the subcycle is contained in $\gamma_1 \cup \gamma_2$, we can regard the subcycle as a linear sequence, according to the clockwise order of its elements along $\gamma_1 \cup \gamma_2$, so the notion of the middle elements is well defined.)

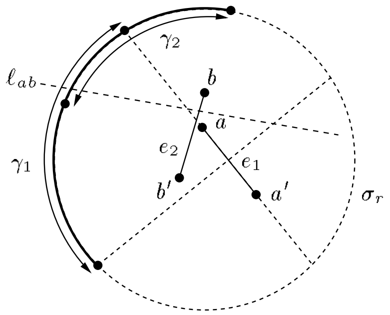

Figure 8: Segments e_1, e_2 and arcs γ_1, γ_2

If γ_1 lies clockwise to γ_2, then all four elements of the subcycle occur within γ_1. But then the perpendicular bisector ℓ_{ab} intersects γ_1 at three points, a contraction. On the other hand, if γ_2 lies clockwise to γ_1, then γ_2 contains the last three elements $\langle b \cdots a \cdots b\rangle$ of our subcycle. This, however, implies that ℓ_{ab} intersects γ_2 twice, which is impossible because the angle spanned by γ_2 is less than $3\pi/4$ and the angle spanned by each of the two arcs into which σ_r is partitioned by ℓ_{ab} is more than $3\pi/4$. This completes the proof of the claim.

A symmetric argument shows that S_R too cannot contain such a subcycle. Hence, if we erase from S_L and from S_R every element equal to its predecessor, we obtain two (cyclic) Davenport–Schinzel sequences of order 2, each composed of at most $2n$ symbols, which are thus of length at most $4n-2$ each. It is now a fairly standard exercise to show that the total length of S is also linear in n (see, e.g., [63]). This completes the proof. $\square$

For each object $c_i \in \mathcal{C}$, define a bivariate function $f_i(\mathbf{x}) = -\min_{q \in c_i} d_P(q, \mathbf{x})$, for $\mathbf{x} \in \mathbb{R}^2$. Let $\mathcal{F} = \{f_i \mid 1 \le i \le n\}$. The minimization diagram of $\mathcal{F}$ is the farthest-neighbor Voronoi diagram of $\mathcal{C}$ under the distance function d_P. Following the same argument as in Theorem 6.1 and using Theorem 9.1, we obtain:

Corollary 9.2 *Let $\mathcal{C}$ be a set of n (possibly intersecting) simply-shaped, compact, convex sets in the plane and d_P be a convex distance function as above. $\mathcal{C}$ can be preprocessed into a farthest-neighbor-searching data structure of size $O(n^{1+\varepsilon})$ so that a query can be answered in $O(\log n)$ time and an object (of the same form) can be inserted into or deleted from $\mathcal{C}$ in $O(n^{\varepsilon})$ time.*

9.2 Maintaining the smallest stabbing disk

Returning to the smallest stabbing disk problem, we need to compute the highest point w^* on the lower envelope $E_{\mathcal{F}}$ of $\mathcal{F}$. In view of Corollary 9.2 and the easily-established fact that the cell of $\mathcal{A}(\mathcal{F})$ lying below $E_{\mathcal{F}}$ is convex, a natural approach to computing w^* is to use the multi-dimensional parametric searching technique of Toledo [66], in a manner similar to that described in the preceding section. Omitting further details, which can easily be worked out by the reader, we conclude:

Theorem 9.3 *A set $\mathcal{C}$ of n (possibly intersecting) simply-shaped, compact, convex sets in the plane can be stored in a data structure of size $O(n^{1+\varepsilon})$ so that a smallest homothetic placement of some given simply-shaped, compact, strictly-convex set P that intersects all objects of $\mathcal{C}$ can be computed in $O(n^{\varepsilon})$ time, after each insertion or deletion of a set into/from $\mathcal{C}$.*

Remark 9.4 A related problem is that of maintaining the smallest disk (or homothetic copy of some given P) *enclosing* all but at most k objects of a dynamically changing collection $\mathcal{C}$. The same technique as above can be used, except that now we need to replace $Vor_P(\mathcal{C})$ by a different farthest-neighbor diagram, where the d_P-distance to an object c is the maximum distance to a point of c. See [65] for a recent study of such Voronoi diagrams.

10 Other Applications

In this section, we list some other applications of the techniques developed in this paper. To keep the length of the paper under control, we omit most of the details concerning these applications; some of these details are fairly routine, but some are more technical and problem-specific. These details will be provided in [32]. We are confident that the new techniques will also find many additional applications.

10.1 Smallest stabbing disk with at most k violations

Let $\mathcal{C}$ and P be as in Section 9, and let $1 \le k \le n$ be an integer parameter. We consider the problem of finding a smallest homothetic placement of P that stabs at least $n-k$ of the objects of $\mathcal{C}$. This problem extends the problem studied by Matoušek [50], in which one seeks the smallest disk stabbing all but at most k points of a given set of points in the plane.

Let $\mathcal{F} = \{f_i \mid 1 \le i \le n\}$ be the set of bivariate functions as defined in Section 9. As easily seen, our goal is to find a point in $\mathcal{A}_{\le k}(\mathcal{F})$ with the maximum z-coordinate. In view of Corollary 9.2, this can be done in time $O(n^{1+\varepsilon}k^2)$, by explicitly computing $\mathcal{A}_{\le k}(\mathcal{F})$, as in Theorem 5.1. However, one can do much better. Note that the following problem

$$\max \quad z, \quad \text{subject to}$$
$$z \le f_i(x, y), \qquad i = 1, \ldots, n$$

is an LP-type problem (see [53, 64]). Matoušek [50, Theorem 2.2] showed that the number of local maxima in the first k levels of $\mathcal{A}(\mathcal{F})$ is $O((k+1)^3)$. Note that each local maxima w of the j-th level lies on at most three function graphs, and is the unique maximum of the lower envelope of a subcollection $\mathcal{F}'$ of $n-j$ functions of $\mathcal{F}$. Moreover, there exists a local maximum w' of the $(j-1)$-st level that 'hides' w, in the sense that w' is the unique maximum of the lower envelope of $\mathcal{F}' \cup \{f\}$, where f is one of the functions whose graph contains w'.

Matoušek also proposed a method for computing all these maxima, which proceeds in a depth-first-search fashion, starting from the unique maximum p in $\mathcal{A}_{\le 0}(\mathcal{F})$ (i.e., the lower envelope of $\mathcal{F}$). Suppose the algorithm is currently at a local maxima q of j-th level for some $j \le k$; q is the unqiue maximum of the lower envelope of a subset $\mathcal{F}' \subseteq \mathcal{F}$ of $n-j$ functions. If $j < k$, the algorithm removes one of the (at most) three constraints defining q, say f, and finds the maximum q' of the LP-type problem defined by the constraints in $\mathcal{F}' \setminus \{f\}$, using an appropriate dynamic data structure. Applying this step repeatedly, we find all local maxima violating at most k constraints. The cost of this algorithm is proportional to the preprocessing cost for constructing the data structure plus the number of maxima (i.e., $O((k+1)^3)$) times the cost of a query and an update of the structure.

In out setting, a point q in $\mathbb{R}^3$ represents a homothetic copy of P that misses exactly k

objects of $\mathcal{C}$ if and only if it lies at the k-th level of $\mathcal{A}(\mathcal{F})$. Hence, to implement Matoušek's technique for our problem, we first construct the data structure of Theorem 9.3, in time $O(n^{1+\varepsilon})$. This also yields a smallest homothetic copy of P that stabs all the objects of $\mathcal{C}$. Then we compute each of the $O((k+1)^3)$ maxima of $\mathcal{A}_{\leq k}(\mathcal{F})$, using the above technique. Each step of the algorithm starts at some local minimum q on one of the first k levels, so that the data structure currently represents all the objects of $\mathcal{C}$ not violating q. It then deletes one of the (at most 3) objects defining q, and queries the structure to obtain the minimum for the resulting set of remaining constraints (as in Theorem 9.3). The cost of such a query is $O(n^{\varepsilon})$. We thus obtain:

Theorem 10.1 *Let $\mathcal{C}$ be a family of n (possibly intersecting) simply-shaped, compact, convex sets in the plane, and let $0 \leq k \leq n$ be a parameter. We can find a smallest homothetic placement of some given simply-shaped, compact, strictly-convex set P, which stabs all except at most k of the sets of $\mathcal{C}$, in time $O(n^{1+\varepsilon} + k^3 n^{\varepsilon})$.*

Remark 10.2 Note that this approach can be generalized to many other optimization problems, where the optimizing object has three degrees of freedom and we seek an optimum solution that violates at most k of the given constraints. The machinery developed in this paper is powerful enough to allow such extensions, requiring only that the lower envelope of any subcollection of objects is a convex surface and that the relevant constraints satisfy some rather mild conditions.

10.2 Segment center with at most k violations

Let S be a set of n points in the plane, and let e be a fixed-length segment. A placement e^* of e is called a *segment center* of S if the maximum distance between the points of S and e^* is minimized. We call e^* a *segment center* of S *with k violations* if the $(k+1)$-st largest distance between e^* and the points of S is minimized. The problem of computing the segment center (without violations) has been studied in [3, 34, 42]; the best algorithm, given in [34], computes the segment center in $O(n^{1+\varepsilon})$ time.

Applying the machinery developed in Section 5, and extending the result of [34], one can obtain the following result.

Theorem 10.3 *The segment center with k violations of a set S of n points in the plane can be computed in time $O(n^{1+\varepsilon}k^2)$.*

Here is a brief sketch of the proof. We first solve the *fixed size problem*: Given a real $d > 0$, determine whether there exists a placement of e for which the $(k+1)$-st largest distance to the points of S is $\leq d$. As follows from the analysis of [34], this is equivalent to the problem of determining whether there exists a (translated and rotated) placement Z of the *hippodrome* $H(e, d)$, defined as the Minkowski sum of e and a (closed) disk of radius d, that contains all but at most k points of S.

Extending the technique of [34], one can reduce this latter problem to the following one: We are given two collections $\mathcal{F}$ and $\mathcal{G}$ of n partially-defined fixed-degree algebraic bivariate functions, and we wish to determine whether there exists an intersection between the first k lower levels of $\mathcal{A}(\mathcal{F})$ and the first k upper levels of $\mathcal{A}(\mathcal{G})$, so that the sum of the levels of such an intersection does not exceed k. This can be solved efficiently using the machinery developed in Section 5. As shown in [34], we have $\psi(n) = O(n \log n)$ for the collections $\mathcal{F}$ and $\mathcal{G}$, which then implies that the asymptotic running time of the algorithm is $O(n^{1+\varepsilon}k^2)$.

Once the fixed-size problem is solved, the segment center problem can be solved by an application of the parametric searching technique of [54]. The bound on the running time remains asymptotically the same.

11 Conclusions

In this paper we have extended known range searching techniques to arrangements of low-degree algebraic bivariate functions in 3-space. By establishing sharp bounds on the vertical decomposition of the first k levels of such an arrangement, we were able to extend the construction of [51] of shallow cuttings to such arrangements. This is turn has yielded a collection of static and dynamic range searching techniques for problems that involve such arrangements, from which we have obtained new and efficient solutions to several geometric optimization problems, including minimum weight Euclidean bipartite matching.

The main open problem that the paper raises is to extend our results to higher dimensions. The first obstacle that we face here is the lack of really sharp bounds on the complexity of vertical decompositions in arrangements of surfaces, even for the whole arrangement, which prevents our technique from 'taking off' at all.

Another open problem is to extend our results to other cases involving shallow levels in 3-dimensional arrangements. For example, we may want to maintain dynamically a single cell in an arrangement of surfaces in $\mathbb{R}^3$, or the union or intersection of a collection of 3-dimensional objects, etc. Extending the analysis of Section 2 to these situations seem considerably more difficult.

References

[1] P. Agarwal, M. de Berg, J. Matoušek, and O. Schwarzkopf, Computing levels in arrangements and higher order Voronoi diagrams, *Proc. 10th Annual Symp. Computational Geometry*, 1994, 67–75.

[2] P. Agarwal, H. Edelsbrunner, O. Schwarzkopf, and E. Welzl, Euclidean minimum spanning trees and bichromatic closest pairs, *Discrete Comput. Geom.* 6 (1991), 407–422.

[3] P. Agarwal, A. Efrat, M. Sharir, and S. Toledo, Computing a segment-center for a planar point set, *J. Algorithms* 15 (1993), 314–323.

[4] P. Agarwal and J. Matoušek, Ray shooting and parametric search, *SIAM J. Comput.* 22 (1993), 794–806.

[5] P. Agarwal and J. Matoušek, Range searching with semialgebraic sets, *Discrete Comput. Geom.* 11 (1994), 393–418.

[6] P. Agarwal and J. Matoušek, Dynamic half-space range searching and its applications, *Algorithmica* 14 (1995), 325–345.

[7] P. Agarwal, J. Matoušek, and O. Schwarzkopf, Computing many faces in arrangements of lines and segments, *Proc. 10th Annual Symp. Computational Geometry*, 1994, 76–84.

[8] P. Agarwal, O. Schwarzkopf, and M. Sharir, Overlay of lower envelopes in three dimensions and its applications, *Discrete Comput. Geom.* 14 (1995), in press.

[9] P. Agarwal, M. Sharir, and P. Shor, Sharp upper and lower bounds for the length of general Davenport Schinzel sequences, *J. of Combin. Theory, Ser. A* 52 (1989), 228–274.

[10] A. Aggarwal, A. Bar-Noy, S. Khuller, D. Kravets, and B. Schieber, Efficient minimum cost matching using quadrangle inequality, *J. Algorithms* 19 (1995), 116–143.

[11] A. Aggarwal, M. Hansen, and T. Leighton, Solving query-retrieval problems by compacting Voronoi diagrams, *Proc. 22nd Annual ACM Symp. Theory of Computing*, 1990, 331–340.

[12] D. Arnon, G. Collins, and S. McCallum, Cylindrical algebraic decomposition I: The basic algorithm, *SIAM J. Comput.* 13 (1984), 865–877.

[13] D. Atkinson and P. Vaidya, Using geometry to solve the transportation problem in the plane, *Algorithmica* 13 (1995), 442–461.

[14] F. Aurenhammer, F. Hoffmann, and B. Aronov, Minkowski-type theorems and least square partitioning, *Proc. 8th Annual Symp. Computational Geometry*, 1992, 350–357.

[15] J. Bentley and J. Saxe, Decomposable searching problems I: Static-to-dynamic transformation, *J. Algorithms* 1 (1980), 301–358.

[16] H. Brönnimann, B. Chazelle, and J. Matoušek, Product range spaces, sensitive sampling, and derandomization, *Proc. 34th Annual IEEE Symp. Foundations of Computer Science*, 1993, 400–409.

[17] S. Buss and P. Yianilos, Linear and $O(n \log n)$ time minimum-cost matching algorithms for quasi-convex tours, *Proc. 5th Annual ACM-SIAM Symp. Discrete Algorithms*, 1994, 65–76.

[18] B. Chazelle, H. Edelsbrunner, L. Guibas, and M. Sharir, A singly exponential stratification scheme for real semialgebraic varieties and its applications, *Proc. 16th Int. Colloq. Automata, Languages and Programming*, 1989, 179–193. (See also *Theoretical Comput. Sci.* 84 (1991), 77–105.)

[19] B. Chazelle, H. Edelsbrunner, L. Guibas, and M. Sharir, Diameter, width, closest line pair, and parametric searching, *Discrete Comput. Geom.* 10 (1993), 183–196.

[20] B. Chazelle and J. Friedman, A deterministic view of random sampling and its use in geometry, *Combinatorica* 10 (1990), 229–249.

[21] B. Chazelle and F. P. Preparata, Halfspace range searching: An algorithmic application of k-sets, *Discrete Comput. Geom.* 1 (1986), 83–93.

[22] B. Chazelle, M. Sharir, and E. Welzl, Quasi-optimal upper bounds for simplex range searching and new zone theorems, *Algorithmica* 8 (1992), 407–430.

[23] K. Clarkson, New applications of random sampling in computational geometry, *Discrete Comput. Geom.* 2 (1987), 195–222.

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

[25] K. Clarkson and P. Shor, Applications of random sampling in computational geometry, II, *Discrete Comput. Geom.* 4 (1989), 387–421.

[26] G. Collins, Quantifier elimination for real closed fields by cylindrical algebraic decomposition, *Second GI Conf. on Automata Theory and Formal Languages*, *Lecture Notes in Computer Science*, vol. 33 (H. Barkhage, ed.), 1975, Springer-Verlag, New York–Berlin–Heidelberg, 134–183.

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

[28] M. de Berg, D. Halperin, and L. Guibas, Vertical decomposition for triangles in 3-space, *Discrete Comput. Geom.*, to appear.

[29] Z. Drezner, The planar two-center and two-median problems, *Transportation Science* 18 (1984), 351–361.

[30] H. Edelsbrunner, *Algorithms in Combinatorial Geometry*, Springer–Verlag, Heidelberg, 1987.

[31] H. Edelsbrunner and R. Seidel, Voronoi diagrams and arrangements, *Discrete Comput. Geom.* 1 (1986), 25-44.

[32] A. Efrat, *Location Problems in Geometric Optimization*, Ph.D. Dissertation, Tel Aviv University, in preparation.

[33] A. Efrat and A. Itai, Improvements on bottleneck matching and related problems using geometry, manuscript, 1995.

[34] A. Efrat and M. Sharir, A near-linear algorithm for the planar segment center problem, to appear in *Discrete Comput. Geom.*

[35] D. Eppstein, Dynamic Euclidean minimum spanning trees and extrema of binary functions, *Discrete Comput. Geom.* 13 (1995), 111–122.

[36] D. Eppstein, Z. Galil, G. Italiano, and A. Nissenzweig, Sparsification—A technique for speeding up dynamic graph algorithms, *Proc. 33rd Annual IEEE Symp. Foundations of Computer Science*, 1992, 60–69.

[37] B. Grünbaum, A proof of Vázsonyi's conjecture, *Bull. Research Council Israel, Sec. A* 6 (1956), 77–78.

[38] L. Guibas, D. Halperin, J. Matoušek, and M. Sharir, Vertical decompsition of arrangements of hyperplanes in four dimensions, *Discrete Comput. Geom.* 14 (1995), 113-122.

[39] D. Halperin and M. Sharir, New bounds for lower envelopes in three dimensions, with applications to visibility of terrains, *Discrete Comput. Geom.* 12 (1994), 313–326.

[40] S. Hart and M. Sharir, Nonlinearity of Davenport-Schinzel sequences and of generalized path compression schemes, *Combinatorica* 6 (1986), 151–177.

[41] D. Haussler and E. Welzl, ϵ-nets and simplex range queries, *Discrete Comput. Geom.* 2 (1987), 127–151.

[42] H. Imai, D.T. Lee, and C. Yang, 1-Segment center covering problems, *ORSA J. Comput.* 4 (1992), 426–434.

[43] D. Leven and M. Sharir, Planning a purely translational motion for a convex object in two-dimensional space using generalized Voronoi diagrams, *Discrete Comput. Geom.* 2 (1987), 9–31.

[44] H. Kuhn, The Hungarian method for the assignment problem, *Naval Research Logistics Quarterly* 2 (1955), 83–97.

[45] E. Lawler, *Combinatorial Optimization: Networks and Matroids*, Saunders College Publishing, Fort Worth, TX, 1976.

[46] D.T. Lee, Two-dimensional Voronoi diagrams in the L_p-metric, *J. Assoc. Comput. Mach.* 27 (1980), 604–618.

[47] D.T. Lee and S. Drysdale, Generalizations of Voronoi diagrams in the plane, *SIAM J. Comput.* 10 (1981), 73–87.

[48] O. Marcotte and S. Suri, Fast matching algorithms for points on a polygon, *SIAM J. Comput.* 20 (1991), 405–422.

[49] J. Matoušek, Approximations and optimal geometric divide-and-conquer, *J. Computer and Systems Sciences* 50 (1995), 203–208.

[50] J. Matoušek, On geometric optimization with few violated constraints, *Discrete Comput. Geom.* 14 (1995), 365–384.

[51] J. Matoušek, Reporting points in halfspaces, *Computational Geometry: Theory and Appl.* 2 (1992), 169–186.

[52] J. Matoušek and O. Schwarzkopf, On ray shooting in convex polytopes, *Discrete Comput. Geom.* 10 (1993), 215–232.

[53] J. Matoušek, M. Sharir, and E. Welzl, A subexponential bound for linear programming and related problems, to appear in *Algorithmica.*

[54] N. Megiddo, Applying parallel computation algorithms in the design of serial algorithms, *J. Assoc. Comput. Mach.* 30 (1983), 852–865.

[55] K. Mehlhorn, S. Meiser, and R. Rasch, Furthest site abstract Voronoi diagrams, Tech. Report, Max-Planck Institut für Informatik, Saarbrücken, 1992.

[56] K. Mulmuley, *Computational Geometry: An Introduction Through Randomized Algorithms*, Prentice Hall, Englewood Cliffs, NJ, 1993.

[57] K. Mulmuley, On levels in arrangements and Voronoi diagrams, *Discrete Comput. Geom.* 6 (1991), 307–338.

[58] M. Overmars, *The Design of Dynamic Data Structures*, Lect. Notes in Computer Science, vol. 156, Springer-Verlag, Berlin, 1983.

[59] J. Schwartz and M. Sharir, On the 'piano movers' problem II: General technique for computing topological properties of real algebraic manifolds, *Adv. Applied Math.* 4 (1983), 298–351.

[60] M. Sharir, Intersection and closest-pair problems for a set of planar discs, *SIAM J. Comput.* 14 (1985), 448–468.

[61] M. Sharir, On k-sets in arrangements of curves and surfaces, *Discrete Comput. Geom.* 6 (1991), 593–613.

[62] M. Sharir, Almost tight upper bounds for lower envelopes in higher dimensions, *Discrete Comput. Geom.* 12 (1994), 327–345.

[63] M. Sharir and P. Agarwal, *Davenport-Schinzel Sequences and Their Geometric Applications*, Cambridge University Press, New York, 1995.

[64] M. Sharir and E. Welzl, A combinatorial bound for linear programming and related problems, *Proc. 9th Annual Symp. Theoretical Aspects of Computer Science*, 1992, pp. 569–579.

[65] M. Sharir and E. Welzl, Circular and spherical seperability, manuscript.

[66] S. Toledo, Maximizing non-linear concave functions in fixed dimensions, *Proc. 33rd Annual IEEE Symp. Foundations of Computer Science*, 1992, 676–685.

[67] P. Vaidya, Geometry helps in matching, *SIAM J. Comput.* 18 (1989), 1201–1225.

[68] L. Valiant, Parallelism in comparison problems, *SIAM J. Comput.* 4 (1975), 348–355.