Low-Dimensional Linear Programming with Violations*

Timothy M. Chan†

October 21, 2004

Abstract

Two decades ago, Megiddo and Dyer showed that linear programming in two and three dimensions (and subsequently any constant number of dimensions) can be solved in linear time. In this paper, we consider the linear programming problem with at most k *violations*, i.e., finding a point inside all but at most k halfspaces, given a set of n halfspaces. We present a simple algorithm in 2-d that runs in $O((n+k^2)\log n)$ expected time; this is faster than earlier algorithms by Everett, Robert, and van Kreveld (1993) and Matoušek (1994) for many values of k, and is probably near-optimal. An extension of our algorithm in 3-d runs in near $O(n+k^{11/4}n^{1/4})$ expected time. Interestingly, the idea is based on concave-chain decompositions (or covers) of the $(\leq k)$-level, previously used in proving combinatorial k-level bounds.

Applications in the plane include improved algorithms for finding a line that misclassifies the fewest among a set of bichromatic points, and finding the smallest circle enclosing all but k points. We also discuss related problems of finding local minima in levels.

Key words. computational geometry, algorithms, linear programming

AMS subject classifications. 68U05, 68Q25, 68W20, 90C05

Abbreviated title. Linear Programming with Violations

1 Introduction

Motivation: outliers in geometric optimization. Consider the following formulation of a line-fitting problem, well-known in computational geometry: given a set of data points (x_1, y_1), ..., (x_n, y_n) in the plane, find a line that minimizes its largest vertical distance to the points. The problem is equivalent to a linear program (LP) in 3 variables (the slope m, the intercept b, and the tolerance δ), where the goal is to minimize δ subject to the constraints $-\delta \leq mx_i + b - y_i \leq \delta$ for $i = 1, \ldots, n$. By known methods [22, 30, 49, 50, 57, 60], the problem can thus be solved in $O(n)$ time.

Though efficient methods exist, whether this formulation is the "right" one for real applications is debatable, as the presence of an occasional faulty data point (a so-called outlier) can drastically change the optimum. Statisticians have looked at more robust formulations of the line-fitting problem, but here we try to address the issue of outliers from a different direction by asking the following natural computational questions, keeping the largest vertical distance as the fitness measure: Given

*A preliminary version of this paper appeared in *Proc. 43rd IEEE Sympos. Found. Comput. Sci.*, 2002. This work was supported in part by an NSERC Research Grant.

†School of Computer Science, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada (tmchan@uwaterloo.ca).

a small integer $k \leq n$, how can we find the line that best fits all but k of the given points? Or, given a prescribed tolerance δ, how can we find the smallest integer k such that a line fitting all but k points exists?

Low-dimensional LP-type techniques can solve other common optimization problems. For example, finding the smallest enclosing circle of a planar point set (the standard 1-center problem) is an instance of 3-d convex programming, and finding the smallest-area enclosing annulus of a planar point set (a circle-fitting problem) is an instance of 4-d linear programming. Due to applications in statistical analysis and computational metrology, it is again natural to consider generalizations of these optimization problems that allow a small number k of violations.

LP with violations: the problem and background. We can define the problem of *linear programming with at most k violations* in d dimensions as follows:

> Given a set H of n closed halfspaces in $\mathbb{R}^d$, a linear objective function f, and a number $0 \leq k < n/2$, we want to minimize f over the region
>
> $$I_k(H) = \{q \in \mathbb{R}^d \mid q \text{ lies outside at most } k \text{ halfspaces of } H\},$$
>
> or report that $I_k(H) = \emptyset$.

Since our interest is in geometric applications, we confine our discussion to small constant values of d. The problem for arbitrary dimensions is NP-complete (for example, see [7, 8]).

Note that $I_0(H)$, the intersection of all halfspaces, is the feasible region of the original LP problem. Following Matoušek [47], we call the special case of the problem in which $I_0(H) \neq \emptyset$ the *feasible* case. In the feasible case (where by an affine transformation, we may assume that the halfspaces are all lower halfspaces or all upper halfspaces), the boundary of $I_k(H)$ is commonly called the *k-level*, and the 0-level, 1-level, ..., k-level collectively form the *$(\leq k)$-level*.

The simplest case, the 2-d feasible problem, can be solved in near-linear time by a parametric or binary search [47, 56]; the current best time bound was obtained by this author [14] using randomization and almost matched the 1-d $O(n + k \log k)$ bound.

The next simplest case, the general 2-d problem, already baffled researchers. Everett, Robert, and van Kreveld [37] first investigated the problem in 1993; they proposed a simple approach that effectively explores the entire solution space, by constructing the $(\leq k)$-level of the upper halfplanes and the $(\leq k)$-level of the lower halfplanes, and "intersecting" the two structures, so to speak. As the $(\leq k)$-level has $O(nk)$ complexity [6, 40] in the plane, they solved the 2-d problem in $O(n \log n + nk)$ time. The approach can probably (though nontrivially) be extended to any fixed dimension, but the running time would be higher, since the $(\leq k)$-level has worst-case complexity $\Theta(n^{\lfloor d/2 \rfloor} k^{\lceil d/2 \rceil})$ in $\mathbb{R}^d$ [24] (in 3-d, a near-$O(nk^2)$ algorithm was indeed obtained by Efrat *et al.* [34]).

Shortly after in 1994, Matoušek [47] proposed another simple approach; it works quite differently and is best described in the feasible case. The algorithm enumerates all local minima of $I_0(H), \ldots, I_k(H)$ by computing the minimum of $I_0(H)$ by LP and repeatedly removing a solution's defining halfspace and reoptimizing, to generate $I_1(H)$ minima from the $I_0(H)$ minimum, $I_2(H)$ minima from the $I_1(H)$ minima, and so on. As the $(\leq k)$-level has $O(k^d)$ local minima [51], the cost of the algorithm is dominated by the cost of $O(k^d)$ dynamic LP operations. The general 2-d problem can be "lifted" to a feasible 3-d problem, and with the appropriate data structures [53], these $O(k^3)$ operations can be carried out in $O(n \log n + k^3 \log^2 n)$ time. The first term has been

problem	best previous result(s)	(refs.)	new result
2-d feasible	$O(n + k^{1-\varepsilon} n^{\varepsilon} \log n)$	[56, 14]	
2-d general	$O(n \log n + nk)$	[37]	
	$O(n \log k + k^3 \log^2 n)$	[47, 11]	$O((n + k^2) \log n)$
3-d feasible	$O(n \log k + k^3 n^{\varepsilon})$	[47, 11]	$O(n \log n + k^2 \log^2 n)$
3-d general	$O(nk^2(\log n + \log^2(n/k)))$	[34]	
	$O(n \log k + k^4 n^{\varepsilon})$	[47, 11]	$O(n \log n + k^{11/4} n^{1/4} \log^c n)$
4-d feasible	$O(n \log k + k^{8/3} n^{2/3+\varepsilon} + k^4 n^{1/3+\varepsilon})$	[47, 11]	$O(n \log n + k^{11/4} n^{1/4} \log^c n)$

Table 1: Time bounds for LP with at most k violations. (In this paper, $\varepsilon > 0$ denotes an arbitrarily small constant, and c denotes a specific constant.)

lowered to $O(n \log k)$ by the author [11, 12]; the second term can probably be lowered slightly as well by adopting recent data structures for dynamic convex hulls [16] (though the particular LP queries needed were not explicitly considered in [16]).

Modulo these small improvements, the rough bounds of $O(nk)$ by Everett *et al.* (for large $k \geq \sqrt{n}$) and $O(n+k^3)$ by Matoušek (for small $k \leq \sqrt{n}$) have surprisingly remained the record for the general 2-d problem for about eight years. There seems no natural way to combine the two approaches to get a uniform time bound for all k. Concerning his algorithm, Matoušek [47] wondered whether it is possible to generate the local minima of $I_k(H)$ directly, without going through $I_0(H), I_1(H), \ldots$, since the number of local minima in the k-level is only $O(k^{d-1})$ [23, 52]. In particular, for the general 2-d problem, is it possible to bypass intermediate "infeasible" minima (defined by triples after the lifting) and generate only solutions (vertices) defined by pairs of halfplanes, as in Everett *et al.*'s algorithm? To put it bluntly, can Matoušek's 2-d algorithm be made more "planar"?

With the status of the 2-d problem unresolved, our understanding of LP with violations in higher dimensions is even more tentative. A suspected lower bound for the general d-dimensional problem (for $k \ll n/2$) is $\Omega(n + k^d)$, because of the following argument: Given a collection of N hyperplanes, the problem of detecting affine degeneracy, i.e., a subset of $d+1$ hyperplanes intersecting at a common point, is conjectured to require $\Omega(N^d)$ time. (See [36] for a proof in a restricted model; for $d = 2$, the problem is so-called 3SUM-*hard* [38].) This problem can be reduced to LP with violations for any $n \geq 2N$ and $k = N - d - 1$: just take the $2N$ lower and upper (closed) halfspaces defined by the N hyperplanes, together with $n - 2N$ copies of the halfspace $x \leq M$ for a sufficiently large M.

Main results. Table 1 summarizes the previous results and our new results. We focus on the general case, since as we will demonstrate in Section 4, the feasible problem in d dimensions reduces to the general problem in $d - 1$ dimensions by parametric or randomized search.

Our main result, presented in Section 2, is an algorithm for the general 2-d problem that runs in $O((n + k^2) \log n)$ expected time (randomization is used in just one step of the algorithm). This algorithm not only simultaneously improves Everett *et al.*'s and Matoušek's algorithms for most values of k (between n^{ε} and $n/\log n$), but also essentially settles the complexity of the 2-d problem up to a logarithmic factor, assuming that the conjectured lower bound holds. The algorithm, like Matoušek's, actually generates all $O(k^2)$ local minima of $I_0(H), \ldots, I_k(H)$. The basic algorithm uses simpler data structures than Matoušek's and is arguably simpler than Everett *et al.*'s as well. The approach is noteworthy: although like Everett *et al.* we will use the $(\leq k)$-level of the lower/upper

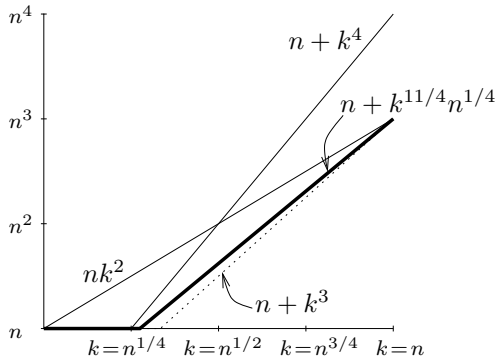

Figure 1: Comparison of time bounds for the general 3-d problem.

halfplanes, we will not build the entire ($\leq k$)-level (which has size $O(nk)$) but instead work with its "concave/convex-chain decomposition" (which involves only a small $O(k)$ number of chains of total size $O(n)$). The concave-chain decomposition idea was instrumental in the recent breakthroughs on k-level complexity and related combinatorial problems [1, 27, 59]; because concave chains and convex polygons enjoy nice computational properties, we demonstrate that this very idea has algorithmic applications as well.

In Section 3, we use similar ideas to obtain an $O(n \log n + k^3 \log^2 n)$ expected time bound for generating an $O(k^3)$-size superset of the local minima of $I_0(H), \ldots, I_k(H)$ in 3-d. The generalization is not trivial, as an analogous concave-surface decomposition of the ($\leq k$)-level in 3-d is not known (but see [43] for a variant); nevertheless, we prove here that a small concave-surface *cover* always exists and can be computed efficiently. Unfortunately, unlike the 2-d algorithm, the 3-d algorithm does not automatically filter out the local minima of $I_0(H), \ldots, I_k(H)$ from the superset. This filtering step turns out to be the computational bottleneck. Sophisticated static data structures for range searching are required to yield our final expected time bound of $O(n \log n + k^{11/4} n^{1/4} \log^c n)$. Although this does not quite approach the suspected lower bound $\Omega(n + k^3)$, it is not far off (see Figure 1), and in particular, improves both Efrat *et al.*'s near-$O(nk^2)$ bound and Matoušek's near-$O(n + k^4)$ bound (the latter requires sophisticated dynamic data structures).

For very small $k = O(n^\varepsilon)$, Matoušek's algorithm can be made to run in $O(n \log k)$ time, as shown by the author [11, 12]. So, by combining two algorithms, we can automatically replace all $\log n$ factors with $\log k$ in the time bounds.

Applications. We can use our algorithms to find the smallest k such that $I_k(H) \neq \emptyset$ within the same time bounds: Given an upper bound $K \geq k$, we can generate all local minima of $I_0(H), \ldots, I_K(H)$ and thus identify the value of k. We can "guess" an upper bound by a standard trick (as in [12]); for example, in 2-d, by trying $K = \sqrt{n}, 2\sqrt{n}, 4\sqrt{n}, \ldots$, the total running time can be bounded by a geometric series and remains $O((n + k^2) \log n)$.

The 2-d algorithm can also be modified for convex programming with nearly the same running time, provided that a linear objective function is used, as explained in the remarks in Section 2. The sample applications below thus follow immediately from these observations (the bounds except the last are probably near-optimal for all $k \ll n/2$).

- Given n red/blue points in the plane, we can find a line ℓ that minimizes k, the total number of red points above ℓ and blue points below ℓ, in $O((n + k^2) \log n)$ expected time. (This was

the original dual problem considered by Everett *et al.* [37].)

- Given n points in the plane and a fixed value δ, we can find a line ℓ that minimizes k, the number of points at a vertical distance exceeding δ from ℓ, in $O((n+k^2)\log n)$ expected time.
- Given n points in the plane and a fixed convex set C of constant complexity, we can translate C to minimize k, the number of points outside C, in $O((n\beta(n)+k^2)\log n)$ expected time, where $\beta(\cdot)$ is a slow-growing, inverse-Ackermann-like function.
- Given n points in the plane and a fixed value δ, we can find an annulus A of area δ that minimizes k, the number of points outside A, in $O(n\log n + k^{11/4}n^{1/4}\log^c n)$ expected time.

We can also use our algorithms to solve feasible LPs with violations in one dimension higher, as explained in Section 4. The applications below follow:

- Given n points in the plane and a number k, we can find the line that minimizes its largest vertical distance to all but k points in $O(n\log n + k^2\log^2 n)$ expected time.
- Given n points in the plane and a number k, we can find the smallest circle enclosing all but k points in $O(n\beta(n)\log n + k^2 n^\varepsilon)$ expected time. (This was the motivating problem considered by Matoušek [47]. Note that in contrast, the related problem of finding the smallest circle enclosing k points is believed to have complexity near $\Theta(nk)$ [26, 35, 42, 46].)
- Given n points in the plane and a number k, we can find the minimum-area annulus enclosing all but k points in $O(n\log n + k^{11/4}n^{1/4}\log^c n)$ expected time.

In the feasible case, several researchers have explored the problem of finding all local minima of the k-level and the $(\le k)$-level.

- By specializing our 2-d algorithm in Section 2 to the feasible case, we can immediately find all $O(k^2)$ local minima of the 2-d $(\le k)$-level in $O((n+k^2)\log n)$ expected time, thus improving the previous $O(n\log n + k^2\log^{3/2} n)$ bound [16, 47].
- As shown in the appendix, we can also find all $O(k)$ local minima of the 2-d k-level in $O((n+(nk)^{3/5})\log^{O(1)} n)$ expected time, improving Katoh and Tokuyama's recent $O((n+(nk)^{2/3})\log^{O(1)} n)$ bound [44].

Again, all $\log n$ factors above can be replaced by $\log k$ by switching to Matoušek's algorithm when $k = O(n^\varepsilon)$.

2 A 2-d algorithm to find all local minima of $I_0(H), \ldots, I_k(H)$

Let L^- and L^+ be the sets of lines bounding the given lower and upper halfplanes respectively. Without loss of generality, assume that the objective is to minimize the x-coordinate. For simplicity, assume also that the input is in general position. The outline of our 2-d algorithm is remarkably simple:

1. Form $O(k)$ concave chains whose union covers (i.e., contains) the $(\le k)$-level of the lower halfplanes; the polygonal chains are made from lines in L^-, are nonoverlapping (i.e., they only intersect at vertices), and have $O(n)$ total size (see Figure 2(a)). Similarly, form $O(k)$ convex chains for the lines in L^+.

2. For each pair of concave and convex chains, compute their left and right intersection points (if exist). Let S be the $O(k^2)$ left intersection points.

3. For each $p \in S$, determine $k^-(p) = \min\{k+1, \text{number of lines of } L^- \text{ strictly below } p\}$ and $k^+(p) = \min\{k+1, \text{number of lines of } L^+ \text{ strictly above } p\}$. If $k^-(p) + k^+(p) \le k$, then report p as a local minimum of $I_{k^-(p)+k^+(p)}(H)$.

To check the correctness of the algorithm, we just have to observe that all local minima of $I_0(H), \ldots, I_k(H)$ are contained in S: each such local minimum lies on both the $(\le k)$-level of the upper halfplanes and the $(\le k)$-level of the lower halfplanes and thus is at the intersection of a concave and a convex chain.

We now explain how to implement the three steps, in order of difficulty, so that the total expected running time is $O((n+k^2)\log n)$.

Step 2: computing S. The intersection of a concave and a convex chain can be found in logarithmic time by known binary-search algorithms (for example, see [20, 28]). So step 2 can be carried out in $O(k^2 \log n)$ time.

Step 3: computing $k^-(p)$ and $k^+(p)$. By symmetry, it suffices to consider the computation of the $k^-(p)$ values. Observe that $k^-(p) = \min\{k+1, \text{number of concave chains strictly below } p\}$: if p is on the $(\le k)$-level of the lower halfplanes, the number of lines of L^- strictly below p is the equal to the number of concave chains strictly below p, since the concave chains cover the $(\le k)$-level and are nonoverlapping; if p is strictly above the $(\le k)$-level, both numbers exceed $k+1$.

Fix a convex chain γ. Each concave chain defines an open interval on γ delimited by the chain's left and right intersection points with γ. (The interval can be half-infinite or empty.) Given a point p lying on γ, a concave chain is strictly below p iff the corresponding interval does not contain p. (See Figure 2(b).) Thus, computing $k^-(p)$ for all points $p \in S$ lying on γ can be reduced to a 1-d counting problem: given $O(k)$ points and $O(k)$ intervals, count how many intervals contain each point. This 1-d problem can be solved in $O(k \log k)$ time by sorting and a linear scan. Repeating the process for each convex chain γ gives $k^-(p)$ for all points $p \in S$ in $O(k^2 \log k)$ time.

Step 1: constructing the concave/convex chains—first option. By symmetry, it suffices to consider the computation of the concave chains. Pick a random integer $k' \in [k, 2k]$. Since the k-level, ..., $2k$-level have combined size $O(nk)$, the k'-level has expected size $O(n)$.

We use a standard idea to decompose the $(\le k')$-level of the lower halfplanes into concave chains (for example, see [1]): Track the $k'+1$ lowest lines of L^- at a vertical sweep line as the sweep line moves from left to right. At $x = -\infty$, our $k'+1$ chains start at the initial $k'+1$ lowest lines. When the $(k'+1)$-st lowest line ℓ_1 and the $(k'+2)$-nd lowest line ℓ_2 are about to switch, the chain currently at ℓ_1 will correspondingly turn right to follow ℓ_2. Switches occur only at concave k'-level vertices, so the $k'+1$ chains thus defined have total expected size $O(n)$ and can be formed in linear time once the k'-level has been constructed. (The example in Figure 2(a) is obtained this way.)

The k'-level can be computed by several output-sensitive algorithms, as surveyed in [15]. The simplest to implement is probably Basch *et al.*'s method using kinetic tournament trees [9]; for an expected $O(n)$-size output, the expected running time is $O(n\alpha(n)\log^2 n)$, where $\alpha(n)$ is the slow-growing inverse Ackermann function. The more complicated, randomized algorithms by Agarwal

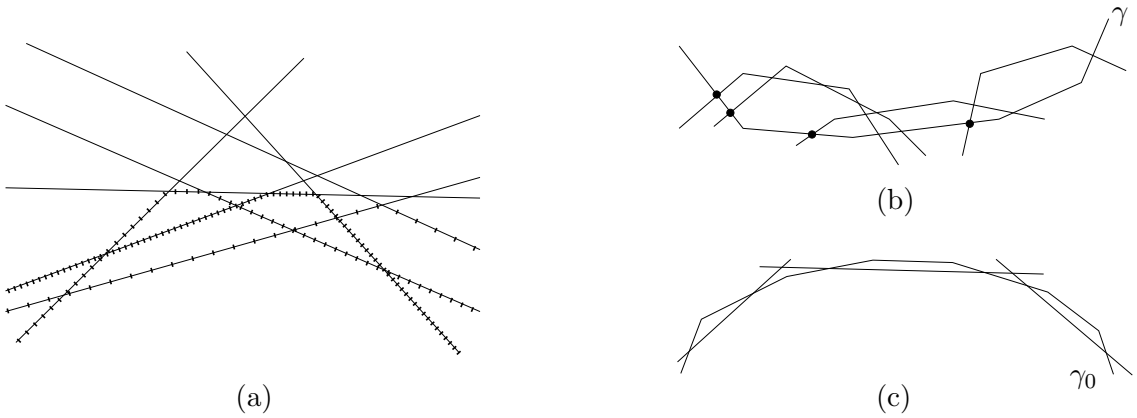

Figure 2: (a) The (≤ 2)-level of a set of lower halfplanes is covered by 3 concave chains (in dotted lines); in this example, the cover happens to be a decomposition. (b) Step 3: Counting concave chains below each point $p \in S$ (in black) on a convex chain γ becomes an interval counting problem. (c) Step 1, second option: Forming concave chains becomes an interval coloring problem.

et al. [2, 15] and Har-Peled [41] are faster, with time bounds of $O(n\alpha(n)^2 \log n)$ and $O(n\alpha(n) \log n)$ respectively. Brodal and Jacob's recent announcement on dynamic convex hulls and kinetic heaps [10] implies the ultimate expected running time of $O(n \log n)$.

Step 1: constructing the concave chains—second option. We now offer a different method for step 1 that also achieves $O(n \log n)$ expected time but has the advantage of being generalizable to 3-d (as we will see in Section 3).

This second option is based on Matoušek's shallow cutting lemma [45]. Ramos [54] (building on [2, 17]) has given an $O(n \log n)$ randomized algorithm to construct such a cutting in 2-d and 3-d. We restate the 2-d result in a form that we find convenient:

Lemma 2.1 *Given n lower halfplanes in $\mathbb{R}^2$, the ($\leq k$)-level can be covered by $O(n/k)$ cells, each intersecting $O(k)$ bounding lines. More specifically, the cells are taken from the vertical decomposition of the region underneath a concave chain γ_0 of size $O(n/k)$. The cells, the list of lines intersecting each cell, and γ_0 can all be constructed in $O(n \log n)$ expected time.*

Proof: Matoušek's shallow cutting lemma [45] and Ramos' algorithm [54] give a set Ξ of cells (triangles) satisfying the first statement in $O(n \log n)$ expected time. To achieve the second statement, we modify the construction. Suppose that the maximum number of lines intersecting a cell is less than bk for some constant b. Remove a cell from Ξ if one of its vertices is above more than $(b+1)k$ lines (because if this is true, the cell does not intersect the ($\leq k$)-level anyway); this condition can be tested in $O(\log n + k)$ time per cell, after preprocessing in $O(n \log n)$ time, by a known data structure for halfplane range reporting queries [21].

Now, let γ_0 be the boundary of the upper hull of the vertices of Ξ, and take the $O(n/k)$ cells (unbounded trapezoids) of its vertical decomposition, which is computable in $O((n/k) \log(n/k))$ time. Clearly, the new cells cover the ($\leq k$)-level. Since a line intersecting a cell must lie below one of its (two) vertices, the list of lines intersecting each new cell has at most $2(b+1)k$ elements and can be generated in $O(\log n + k)$ time per cell, again by halfplane range reporting. □

Say that two lines are *compatible* in a region R if no point in R is above both lines. The problem of constructing the concave chains can be reduced to coloring the lines of L^- so that lines of the same color are compatible in the $(\leq k)$-level of the lower halfplanes (or in any region covering the $(\leq k)$-level): Indeed, nonoverlapping concave chains can be formed by simply taking the lower envelope of lines of each color; the chains cover the $(\leq k)$-level, because given a point p on the $(\leq k)$-level incident to some line ℓ, p cannot be above another line of the same color as ℓ and so must appear in the lower envelope of that color.

To solve this coloring problem, we invoke Lemma 2.1 to get a concave chain γ_0 above the $(\leq k)$-level. Each line of L^- defines an interval (possibly half-infinite or empty) on γ_0 delimited by the line's left and right intersection points with γ_0; each interval can again be computed in logarithmic time by binary search. Two lines are compatible in the region underneath γ_0 iff their corresponding intervals are disjoint. (See Figure 2(c).) Thus, our problem is reduced to coloring of an interval graph.

Observe that the intervals have *depth* $O(k)$, i.e., each point p on γ_0 is contained in at most $O(k)$ intervals: p is contained in an interval iff p lies above the corresponding line, but p is above only $O(k)$ lines since each cell intersects $O(k)$ lines by construction. As is well-known (e.g., see [39]), the chromatic number of an interval graph is equal to the depth (also equal to the clique number), and an optimal coloring can be found in $O(n \log n)$ time by a standard greedy strategy (sequentially coloring the intervals in sorted order of left endpoints). This gives an $O(k)$-coloring, and thus a cover by $O(k)$ concave chains, in $O(n \log n)$ expected time.

We have thus proved:

Theorem 2.2 *For n halfplanes in $\mathbb{R}^2$, linear programming with at most k violations can be solved in $O((n+k)\log n)$ expected time.*

Remarks. The construction [54] used in Lemma 3.1 is fairly involved. A much simpler alternative is to choose a random sample of $\frac{n}{2k}$ bounding planes and take the vertical decomposition of their lower envelope. This construction behaves the same "on average", but is only guaranteed to cover each vertex of the $(\leq k)$-level with constant probability (for example, as observed in [1]). The resulting algorithm would be Monte Carlo.

Degenerate cases can be handled by direct modifications of the algorithm or by general symbolic perturbation techniques [31, 32]. With the latter choice, we need to perturb the lines in L^- upward and the lines in L^+ downward (as also suggested by Matoušek [47]), so that a vertex v (lying on possibly more than two lines) violates at most k constraints iff some point in the neighborhood of v violates at most k constraints after the perturbation.

The algorithm can be modified to work if the constraints H are not halfplanes but convex sets of constant description complexity: For each convex set, we form a concave/convex x-monotone curve by taking its upper/lower boundary and attaching near-vertical downward/upward rays at the two endpoints. We let L^- and L^+ instead be the sets of these concave and convex curves respectively. Step 1 still works using the first option, since many of the k-level algorithms [2, 9, 15, 41] generalize to curves, with $\alpha(n)$ replaced by a similar slow-growing function $\beta(n)$ in the time bound. Step 2 generalizes, since concave/convex chains formed by concave/convex curves are still concave/convex. Step 3 requires a change, though. Given that $k^-(p), k^+(p) \leq k$, the number of constraints violated by p is not necessarily $k^-(p)+k^+(p)$, but is rather $k^-(p)+k^+(p)-k_0(p)$, where $k_0(p)$ = number of convex sets whose x-projection does not contain p. Fortunately, computing the

$k_0(p)$'s is also a 1-d interval counting problem and can be solved by sorting. The total expected running time is $O((n\beta(n) + k^2)\log n)$.

3 A 3-d algorithm to find all local minima of $I_0(H), \ldots, I_k(H)$

Let Π^- and Π^+ be the sets of planes bounding the given lower and upper halfspaces respectively. Our 3-d algorithm proceeds similarly:

1. Form $O(k)$ concave surfaces whose union covers the $(\leq k)$-level of the lower halfspaces; the polyhedral surfaces are made from planes of Π^-, are nonoverlapping (i.e., they only intersect at 0-d or 1-d features), and have $O(n)$ total size. Similarly, form $O(k)$ convex surfaces for the planes of Π^+.

2. For each subset of at most three surfaces, find the minimum point in the intersection of the at most three convex polytopes bounded by these surfaces. Let S be the $O(k^3)$ minima obtained.

3. For each $p \in S$, determine $k^-(p) = \min\{k+1, \text{number of planes of } \Pi^- \text{ strictly below } p\}$ and $k^+(p) = \min\{k+1, \text{number of planes of } \Pi^+ \text{ strictly above } p\}$. If $k^-(p)+k^+(p) \leq k$, then report p as a local minimum of $I_{k^-(p)+k^+(p)}(H)$.

Correctness follows since S contains all local minima of $I_0(H), \ldots, I_k(H)$: each such local minimum p lies on the $(\leq k)$-level of the lower halfspaces and of the upper halfspaces; at the neighborhood of p, its three defining halfspaces are part of up to three concave/convex surfaces; p is the minimum point in the intersection of the corresponding polytopes.

We now explain how to implement each step.

Step 2: computing S. If the polytopes are stored in hierarchical representations [29], which require $O(n)$ preprocessing time, then the minimum in the intersection of three (or less) convex polytopes can be found by an $O(\log^3 n)$ algorithm [33]. If the polytopes are instead stored in drum representations [55], which require $O(n \log n)$ preprocessing time [13], then the minimum can be found by an $O(\log^2 n)$ algorithm [13]. With the latter option, this step takes $O(k^3 \log^2 n)$ time.

Step 1: constructing the concave/convex surfaces—existence proof. By symmetry, it suffices to consider the computation of the concave surfaces. Following the second option in Section 2, we reduce the problem to coloring the planes of Π^- so that two planes of the same color are compatible in the $(\leq k)$-level of the lower halfspaces (i.e., no point in the region is above both planes). We use a 3-d version of Lemma 2.1:

Lemma 3.1 *Given n lower halfspaces in $\mathbb{R}^3$, the $(\leq k)$-level can be covered by $O(n/k)$ cells, each intersecting $O(k)$ bounding planes. More specifically, the cells are taken from the vertical decomposition of the region underneath a polyhedral concave surface γ_0 of size $O(n/k)$. The cells, the list of planes intersecting each cell, and γ_0 can all be constructed in $O(n \log n)$ expected time.*

Proof: As in the proof of Lemma 2.1, we apply Matoušek's shallow cutting lemma and Ramos' algorithm, which work in 3-d. We modify the construction in the same way, this time, using a randomized data structure for 3-d halfspace range reporting [17]. □

To solve the coloring problem, we invoke Lemma 3.1 to get a concave surface γ_0 lying above the $(\leq k)$-level. Each plane of Π^- defines a set on γ_0 formed by intersecting its upper halfspace with γ_0. Two planes are compatible in the region underneath γ_0 iff the corresponding sets are disjoint.

As before, the sets have depth $O(k)$: a point p on γ lies above at most $O(k)$ planes since each cell intersects $O(k)$ planes by construction. We would like to infer that the intersection graph of planar convex sets of depth $O(k)$ is $O(k)$-colorable; however, this is not necessarily true in general. Fortunately, our sets are *pseudo-disks*, i.e., the boundaries of each pair intersect at most twice: the intersection of two planes is a line, and a line intersects γ_0 at most twice. By a lemma of Sharir [58] (which extends a lemma of Pach), a 2-d arrangement of n pseudo-disks has $O(nk)$ intersections if their depth is $O(k)$. It follows that there exists a pseudo-disk that intersects at most $O(k)$ other pseudo-disks. Remove this pseudo-disk and repeat. As a result, we obtain an acyclic orientation of the intersection graph so that the maximum in-degree δ is bounded by $O(k)$ (i.e., the *degeneracy* or *inductiveness* of the graph is $O(k)$). As is well-known (e.g., see [39]), the chromatic number is bounded by $\delta + 1$, and a corresponding coloring can be found by a standard greedy strategy. We conclude that an $O(k)$-coloring of the pseudo-disks, hence, a cover by $O(k)$ concave surfaces, exists.

Step 1: constructing the concave surfaces—algorithmic proof. Further ideas are needed to turn the above proof into an efficient algorithm, as we cannot afford to build the intersection graph of the pseudo-disks and color sequentially. Here, we return to Matoušek's shallow cutting lemma (Lemma 3.1) and avoid Sharir's lemma (both were incidentally proven by random sampling arguments). Recall that we have $a|\Pi^-|/k$ cells, each intersected by at most bk planes, for some constants a and b. Call a plane *heavy* if it intersects more than $2ab$ cells, and *light* otherwise (this is inspired by a similar definition of "bad" and "good" in [4]). At most half of the planes of Π^- are heavy.

First, recursively color the heavy planes from the palette $\{1, \ldots, 4ab^2k\}$ so that planes of the same color are compatible in the $(\leq k)$-level of the corresponding heavy halfspaces (and thus in the $(\leq k)$-level of all lower halfspaces). Stop if the number of planes drops below $4ab^2k$. For each cell, maintain a dictionary of the colors used so far among the planes intersecting the cell.

Now, take each light plane π (in any order). Randomly select a color from $\{1, \ldots, 4ab^2k\}$. Check that the color is not in the dictionary of any of the at most $2ab$ cells π intersects. If so, give π that color and update the dictionaries; otherwise, retry with another random selection. Since each of the at most $2ab$ dictionaries contains at most bk colors, the probability of success in each trial is at least $1/2$, so this process takes expected $O(1)$ dictionary operations for each π. If the plane π and a previous plane are incompatible in the region underneath γ_0, then some cell is intersected by both planes, and their colors are different by construction. So, after all light planes are processed, we have a correct $(4ab^2k)$-coloring.

The expected running time satisfies the recurrence $T(n) = T(n/2) + O(n \log n)$, which solves to $O(n \log n)$.

Step 3: computing $k^-(p)$ and $k^+(p)$. By symmetry, it suffices to consider the computation of the $k^-(p)$ values. The geometry of concave/convex surfaces does not seem to help here, unlike in the 2-d algorithm, and we need to resort to more complicated techniques.

We wish to determine the number of planes strictly below each of $O(k^3)$ points. In dual space, this is the halfspace range counting problem. Since an upper limit $k + 1$ is placed on the counts, one solution is to use known output-sensitive results, as given in [18, Theorem 6], to answer each

halfspace range counting query in $O((1+(nk^2/m)^{1/3})n^{\varepsilon})$ time after preprocessing in $O(mn^{\varepsilon})$ time. By setting the tradeoff parameter $m = \max\{k^{11/4}n^{1/4}, n\}$, the total cost of $O(k^3)$ queries becomes $O(n^{1+\varepsilon} + k^{11/4}n^{1/4+\varepsilon})$.

Alternatively, a more direct solution is to use Lemma 3.1. To compute $k^-(p)$, locate the cell Δ that contains p (by 2-d point location), extract the list of the $O(k)$ planes intersecting Δ, and count the number of planes in this list strictly below p. Recalling that q halfspace range counting queries on a set of size $O(k)$ in 3-d take $O(k \log k + (kq)^{3/4} \log^c k)$ time [3], we obtain the following bound on the total cost, where the q_i's sum to $O(k^3)$:

$$\begin{aligned} O\left(n \log n + \sum_{i=1}^{O(n/k)} (kq_i)^{3/4} \log^c k\right) &= O(n \log n + (n/k)^{1/4}(k \cdot k^3)^{3/4} \log^c k) \\ &= O(n \log n + k^{11/4}n^{1/4} \log^c k). \end{aligned}$$

Since range searching structures are used as a subroutine, the result is admittedly theoretical, but because the queries are off-line, (hierarchical) cuttings instead of partition trees are sufficient [3, 19] and the constant c is small. (Without sophisticated data structures, one can still get a near-$O(n+k^4)$ time bound.)

We have thus proved:

Theorem 3.2 *For n halfspaces in $\mathbb{R}^3$, linear programming with at most k violations can be solved in $O(n \log n + k^{11/4}n^{1/4} \log^{O(1)} n)$ expected time.*

4 On the feasible case

We now solve the feasible LP problem with violations in $\mathbb{R}^d$ by reducing it to general LP with violations in $\mathbb{R}^{d-1}$.

Assume the objective is to minimize the first coordinate x. Let ξ^* be the minimum x-coordinate in $I_k(H)$. First, find some $v_0 \in I_0(H)$ by LP. Since $I_k(H)$ is connected in the feasible case, given ξ smaller than the x-coordinate of v_0, we can decide whether $\xi^* \leq \xi$ by testing whether $I_k(H)$ intersects the vertical hyperplane $x = \xi$, or equivalently, whether $I_k(H_\xi) \neq \emptyset$, where H_ξ is the set of $(d-1)$-d halfspaces formed by intersecting the halfspaces of H with $x = \xi$.

3-d feasible LP with violations. In the $d = 3$ case, the decision problem can therefore be solved by the 2-d algorithm of Section 2 in $O((n+k^2)\log n)$ time. To solve the optimization problem, we can apply parametric search [48]. (We assume that the reader is familiar with this technique; if not, see [5] for more information.) Unfortunately, step 1 of our algorithm, the construction of the concave/convex chains, appears difficult to parallelize efficiently.

To circumvent this difficulty, we preprocess before parametric searching by constructing the 3-d concave/convex surface cover using step 1 of the algorithm of Section 3, in $O(n \log n)$ expected time. If the bounding polytopes are stored in drum representations [55] via persistent search trees, as suggested in [13], we can retrieve a binary-searchable copy of the intersection of each surface with any vertical plane $x = \xi$ in logarithmic time. These 2-d slices form concave/convex chains covering the $(\leq k)$-level of the lower/upper halfplanes of H_ξ. Thus, for any given ξ, we can decide whether $I_k(H_\xi) = \emptyset$ in $O(k^2 \log n)$ time using steps 2 and 3 of the 2-d algorithm. These two steps

are parallelizable in $O(\log n)$ time with $O(k^2)$ processors, by using the AKS sorting network for the interval counting problem in step 3 (the linear scan does not need to be parallelized because no more comparisons with ξ are involved after sorting). Thus, ξ^* can be found by Megiddo's parametric search [48], with Cole's improvement [25], in $O(k^2 \log^2 n)$ time. Including the preprocessing, the entire algorithm takes $O(n \log n + k^2 \log^2 n)$ expected time.

4-d feasible LP with violations. Parametric search is more problematic in the $d = 4$ case. Not only is the decision algorithm (the 3-d algorithm of Section 3) difficult to parallelize (because of step 1), but an efficient global preprocessing in 4-d is not available to remedy the situation. Fortunately, the author's randomized optimization technique [14] is applicable (which uses the decision algorithm only as a black box) and yields the best result in this case.

Lemma 4.1 *If LP with at most k violations in $\mathbb{R}^{d-1}$ can be solved in $O(T(n,k))$ expected time, then feasible LP with at most k violations in $\mathbb{R}^d$ can be solved in $O(T(n,k))$ expected time, provided that $T(n,k)/n^\varepsilon$ is monotone increasing in n.*

The $d = 2$ case of the above lemma is shown in [14, Theorem 5.2]. Since the proof in higher dimensions is essentially identical (using known constant-size cuttings in higher dimensions), we will not repeat the proof here.

By Lemma 4.1 and the algorithm of Section 3, we can thus solve the 4-d feasible case in $O(n \log n + k^{11/4} n^{1/4} \log^c n)$ expected time.

Theorem 4.2 *For n halfspaces in $\mathbb{R}^3$ with a nonempty common intersection, linear programming with at most k violations can be solved in $O(n \log n + k^2 \log^2 n)$ expected time.*

For n halfspaces in $\mathbb{R}^4$ with a nonempty common intersection, linear programming with at most k violations can be solved in $O(n \log n + k^{11/4} n^{1/4} \log^{O(1)} n)$ expected time.

Remark. A similar approach can be used to find the smallest circle enclosing all but k points, given n points $\{(a_i, b_i)\}_i$ in the plane. This problem is a variant of 3-d feasible LP with violations: the objective is now to minimize a convex function x^2+y^2+z, but the constraints $z \geq -2a_i x - 2b_i y + a_i^2 + b_i^2$ are still linear. The same technique for Lemma 4.1 reduces the problem to 2-d convex programming with violations (deciding whether there exists a point lying in all but k planar convex sets), which can be solved in $O((n\beta(n) + k) \log n)$ expected time by the remark in Section 2. The upper bound $T(n) = O(n\beta(n) \log n + kn^\varepsilon)$ can then be used.

(Alternatively, it might be possible to solve this problem by parametric search, but certain details are unclear, so we will not comment further.)

5 Conclusions

We have presented a new, simple approach to solving LPs with violations in two and three dimensions. The running time of our 2-d algorithm ($O((n+k^2) \log n)$) is almost optimal for all values of $k \ll n/2$ under a well-known conjecture; an open question is whether the time bound can be further improved to $O(n \log k + k^2)$. Our algorithm uses $O(n + k^2)$ space; another question is whether the storage requirement can be reduced to $O(n)$.

For the general 3-d problem, there is still a small gap between our time bound and the conjectured $\Omega(n+k^3)$ lower bound. For the general 4-d problem, our approach does not seem to work as well due to

various reasons (for instance, we can no longer afford to construct concave/convex surfaces explicitly, since convex polytopes defined by n hyperplanes in $\mathbb{R}^d$ may have $\Omega(n^{\lfloor d/2 \rfloor})$ size). Determining the complexity of LP with violations in dimension four and beyond thus remains a challenging problem.

Appendix: Finding all local minima of $I_k(H)$ in the 2-d feasible case

In the 2-d feasible case, Katoh and Tokuyama [44] recently showed that all $O(k)$ local y-maxima of the k-level of n lower halfplanes can be enumerated in $O(n \log n + (nk)^{2/3} \log^{O(1)} n)$ time. In this appendix, we note that the time bound can be reduced to $O((n + (nk)^{3/5}) \log^{O(1)} n)$ using randomization. This result is independent of the rest of the paper and of mainly theoretical interest, because in practice the k-level in 2-d tends to have near-linear size, in which case the problem can be solved directly in near-linear time.

First of all, it is no loss of generality in the feasible case to assume that the objective is to maximize the y-coordinate and the halfplanes are all lower halfplanes: the first condition can be met by rotation; for the second condition, we can find a point $v_0 \in I_0(H)$ by LP, make v_0 the origin by translation, and then transform each halfplane $ax + by \leq 1$ to the lower halfplane $y' \leq ax' - b$ by an affine map ($x' = -x/y$, $y' = -1/y$).

Our result is obtained by the following observation (which was also used in an algorithm in [15]): although the current best bound on size of the k-level is $O(nk^{1/3})$ by Dey [27], a random level nearby has lower complexity. Let j be a parameter to be set later. Pick a random integer $j' \in [0, j]$ and construct the $(k - j')$-level $\mathcal{L}^-$ and the $(k + j')$-level $\mathcal{L}^+$. Since the $(k - j)$-level, ..., $(k + j)$-level have combined size $O(nk^{1/3}j^{2/3})$ [27], $\mathcal{L}^-$ and $\mathcal{L}^+$ have expected size $O(n(k/j)^{1/3})$ and can therefore be constructed in $O(n(k/j)^{1/3} \log^{O(1)} n)$ expected time by known output-sensitive algorithms [15].

Among the n bounding lines, track the subset formed by the $(k - j' + 1)$-st, ..., $(k + j' + 1)$-st lowest lines at a vertical sweep line as the sweep line moves from left to right. Changes to the subset occur only at vertices of $\mathcal{L}^-$ and $\mathcal{L}^+$, and thus this process takes $O(n(k/j)^{1/3})$ time.

Now, divide the plane into $m = O(nk^{1/3}/j^{4/3})$ vertical slabs, each containing at most j vertices of $\mathcal{L}^-$ and $\mathcal{L}^+$. Take each slab σ. A line intersecting the k-level within σ must intersect $\mathcal{L}^-$ or $\mathcal{L}^+$, or be among the $(k - j' + 1)$-st, ..., $(k + j' + 1)$-st lowest lines at the left/right wall of σ. Therefore, we can form a list L_σ of $O(j)$ size that contains all lines involved in the k-level within σ. Within σ, the k-level of all lower halfplanes coincides with a level of the $O(j)$ lower halfplanes defined by L_σ. We can apply Katoh and Tokuyama's algorithm to find all k_σ local y-maxima within σ in $O(j \log j + (jk_\sigma)^{2/3} \log^{O(1)} j)$ time, because their algorithm is output-sensitive [44]. The total time over all slabs is given below, up to polylogarithmic factors, where the k_σ's sum to $O(k)$:

$$\begin{aligned} O\left(mj + \sum_\sigma (jk_\sigma)^{2/3}\right) &= O(mj + m^{1/3}j^{2/3}k^{2/3}) \\ &= O(n(k/j)^{1/3} + n^{1/3}k^{7/9}j^{2/9}). \end{aligned}$$

Setting $j = \min\{n^{6/5}/k^{4/5}, k\}$ yields an expected time bound of $O((n + (nk)^{3/5}) \log^{O(1)} n)$.

Remark. If Dey's bound on the combined size of the $(k - j)$-level, ..., $(k + j)$-level can be improved to $O(nk^\alpha j^{1-\alpha})$ for some constant $\alpha > 0$, then the time bound would work out to be $O((n + (nk^{3\alpha})^{1/(1+2\alpha)}) \log^{O(1)} n)$.

References

[1] P. K. Agarwal, B. Aronov, T. M. Chan, and M. Sharir. On levels in arrangements of lines, segments, planes, and triangles. *Discrete Comput. Geom.*, 19:315–331, 1998.

[2] P. K. Agarwal, M. de Berg, J. Matoušek, and O. Schwarzkopf. Constructing levels in arrangements and higher order Voronoi diagrams. *SIAM J. Comput.*, 27:654–667, 1998.

[3] P. K. Agarwal and J. Erickson. Geometric range searching and its relatives. In *Discrete and Computational Geometry: Ten Years Later* (B. Chazelle, J. E. Goodman, and R. Pollack, eds.), AMS Press, pages 1–56, 1999.

[4] P. K. Agarwal and J. Matoušek. Dynamic half-space range reporting and its applications. *Algorithmica*, 13:325–345, 1995.

[5] P. K. Agarwal and M. Sharir. Efficient algorithms for geometric optimization. *ACM Comput. Surv.*, 30:412–458, 1998.

[6] N. Alon and E. Győri. The number of small semispaces of a finite set of points in the plane. *J. Combin. Theory Ser. A*, 41:154–157, 1986.

[7] E. Amaldi and V. Kann. The complexity and approximability of finding maximum feasible subsystems of linear relations. *Theoret. Comput. Sci.*, 147:181–210, 1995.

[8] E. Amaldi and V. Kann. On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems. *Theoret. Comput. Sci.*, 209:237–260, 1998.

[9] J. Basch, L. J. Guibas, and J. Hershberger. Data structures for mobile data. *J. Algorithms*, 31:1–28, 1999.

[10] G. S. Brodal and R. Jacob. Dynamic planar convex hull. In *Proc. 43rd IEEE Sympos. Found. Comput. Sci.*, pages 617–626, 2002.

[11] T. M. Chan. Fixed-dimensional linear programming queries made easy. In *Proc. 12th ACM Sympos. Comput. Geom.*, pages 284–290, 1996.

[12] T. M. Chan. Output-sensitive results on convex hulls, extreme points, and related problems. *Discrete Comput. Geom.*, 16:369–387, 1996.

[13] T. M. Chan. Deterministic algorithms for 2-d convex programming and 3-d online linear programming. *J. Algorithms*, 27:147–166, 1998.

[14] T. M. Chan. Geometric applications of a randomized optimization technique. *Discrete Comput. Geom.*, 22:547–567, 1999.

[15] T. M. Chan. Remarks on k-level algorithms in the plane. Manuscript, 1999.

[16] T. M. Chan. Dynamic planar convex hull operations in near-logarithmic amortized time. *J. ACM*, 48:1–12, 2001.

[17] T. M. Chan. Random sampling, halfspace range reporting, and construction of ($\leq k$)-levels in three dimensions. *SIAM J. Comput.*, 30:561–575, 2001.

[18] T. M. Chan. On enumerating and selecting distances. *Int. J. Comput. Geom. Appl.*, 11:291–304, 2001.

[19] B. Chazelle. Cutting hyperplanes for divide-and-conquer. *Discrete Comput. Geom.*, 9:145–158, 1993.

[20] B. Chazelle and D. P. Dobkin. Intersection of convex objects in two and three dimensions. *J. ACM*, 34:1–27, 1987.

[21] B. Chazelle, L. Guibas, and D. T. Lee. The power of geometric duality. *BIT*, 25:76–90, 1985.

[22] K. L. Clarkson. Las Vegas algorithms for linear and integer programming when the dimension is small. *J. ACM*, 42:488–499, 1995.

[23] K. L. Clarkson. A bound on local minima of arrangements that implies the upper bound theorem. *Discrete Comput. Geom.*, 10:427–233, 1993.

[24] K. L. Clarkson and P. W. Shor. Applications of random sampling in computational geometry, II. *Discrete Comput. Geom.*, 4:387–421, 1989.

[25] R. Cole. Slowing down sorting networks to obtain faster sorting algorithms. *J. ACM*, 34:200–208, 1987.

[26] A. Datta, H.-P. Lenhof, C. Schwarz, and M. Smid. Static and dynamic algorithms for k-point clustering problems. *J. Algorithms*, 19:474–503, 1995.

[27] T. K. Dey. Improved bounds on planar k-sets and k-levels. *Discrete Comput. Geom.*, 19:373–382, 1998.

[28] D. P. Dobkin and D. G. Kirkpatrick. Fast detection of polyhedral intersection. *Theoret. Comput. Sci.*, 27:241–253, 1983.

[29] D. P. Dobkin and D. G. Kirkpatrick. Determining the separation of preprocessed polyhedra: a unified approach. In *Proc. 17th Int. Colloq. Automata, Languages, and Programming*, Lect. Notes in Comput. Sci., vol. 443, Springer-Verlag, pages 400–413, 1990.

[30] M. E. Dyer. Linear time algorithms for two- and three-variable linear programs. *SIAM J. Comput.*, 13:31–45, 1984.

[31] H. Edelsbrunner and E. P. Mücke. Simulation of simplicity: a technique to cope with degenerate cases in geometric algorithms. *ACM Trans. Graphics*, 9:66–104, 1990.

[32] I. Emiris and J. Canny. A general approach to removing degeneracies. *SIAM J. Comput.*, 24:650–664, 1995.

[33] D. Eppstein. Dynamic three-dimensional linear programming. *ORSA J. Comput.*, 4:360–368, 1992.

[34] A. Efrat, M. Lindenbaum, and M. Sharir. Finding maximally consistent sets of halfspaces. In *Proc. 5th Canad. Conf. Comput. Geom.*, pages 432–436, 1993.

[35] A. Efrat, M. Sharir, and A. Ziv. Computing the smallest k-enclosing circle and related problems. *Comput. Geom. Theory Appl.*, 4:119–136, 1994.

[36] J. Erickson. New lower bounds for convex hull problems in odd dimensions. *SIAM J. Comput.*, 28:1198–1214, 1999.

[37] H. Everett, J.-M. Robert, and M. van Kreveld. An optimal algorithm for the $(\leq k)$-levels, with applications to separation and transversal problems. *Int. J. Comput. Geom. Appl.*, 6:247–261, 1996.

[38] A. Gajentaan and M. H. Overmars. On a class of $O(n^2)$ problems in computational geometry. *Comput. Geom. Theory Appl.*, 5:165–185, 1995.

[39] M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs.* Academic Press, New York, 1980.

[40] J. E. Goodman and R. Pollack. On the number of k-subsets of a set of n points in the plane. *J. Combin. Theory Ser. A*, 36:101–104, 1984.

[41] S. Har-Peled. Taking a walk in a planar arrangement. *SIAM J. Comput.*, 30:1341–1367, 2000.

[42] S. Har-Peled and S. Mazumdar. Fast algorithms for computing the smallest k-enclosing disc. In *Proc. 11th European Sympos. Algorithms*, Lect. Notes in Comput. Sci., vol. 2832, Springer-Verlag, pages 278–288, 2003. *Algorithmica*, to appear.

[43] N. Katoh and T. Tokuyama. Lovász's lemma for the three-dimensional k-level of concave surfaces and its applications. *Discrete Comput. Geom.*, 27:567–584, 2002.

[44] N. Katoh and T. Tokuyama. Notes on computing peaks in k-levels and parametric spanning trees. In *Proc. 17th ACM Sympos. Comput. Geom.*, pages 241–248, 2001.

[45] J. Matoušek. Reporting points in halfspaces. *Comput. Geom. Theory Appl.*, 2:169–186, 1992.

[46] J. Matoušek. On enclosing k points by a circle. *Inform. Process. Lett.*, 53:217–221, 1995.

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

[48] N. Megiddo. Applying parallel computation algorithms in the design of serial algorithms. *J. ACM*, 30:852–865, 1983.

[49] N. Megiddo. Linear time algorithms for linear programming in R^3 and related problems. *SIAM J. Comput.*, 12:759–776, 1983.

[50] N. Megiddo. Linear programming in linear time when the dimension is fixed. *J. ACM*, 31:114–127, 1984.

[51] K. Mulmuley. Output sensitive construction of levels and Voronoi diagrams in R^d of order 1 to k. In *Proc. 22nd ACM Sympos. Theory Comput.*, pages 322–330, 1990.

[52] K. Mulmuley. Dehn-Sommerville relations, upper bound theorem, and levels in arrangements. In *Proc. 9th ACM Sympos. Comput. Geom.*, pages 240–246, 1993.

[53] M. H. Overmars and J. van Leeuwen. Maintenance of configurations in the plane. *J. Comput. Sys. Sci.*, 23:166–204, 1981.

[54] E. Ramos. On range reporting, ray shooting, and k-level construction. In *Proc. 15th ACM Sympos. Comput. Geom.*, pages 390–399, 1999.

[55] M. Reichling. On the detection of a common intersection of k convex polyhedra. In *Computational Geometry and its Applications*, Lect. Notes in Comput. Sci., vol. 333, Springer-Verlag, pages 180–186, 1988.

[56] T. Roos and P. Widmayer. k-violation linear programming. *Inform. Process. Lett.*, 52:109–114, 1994.

[57] R. Seidel. Small-dimensional linear programming and convex hulls made easy. *Discrete Comput. Geom.*, 6:423–434, 1991.

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

[59] M. Sharir, S. Smorodinsky, and G. Tardos. An improved bound for k-sets in three dimensions. *Discrete Comput. Geom.*, 26:195–204, 2001.

[60] M. Sharir and E. Welzl. A combinatorial bound for linear programming and related problems. In *Proc. 9th Sympos. Theoret. Aspects Comput. Sci.*, Lect. Notes in Comput. Sci., vol. 577, Springer-Verlag, pages 569–579, 1992.