Discrete Comput Geom 14:365–384 (1995)

Discrete & Computational Geometry

On Geometric Optimization with Few Violated Constraints*

J. Matoušek

Department of Applied Mathematics, Charles University,
Malostranské nám. 25, 118 00 Praha 1, Czech Republic
matousek@kam.mff.cuni.cz

Abstract. We investigate the problem of finding the best solution satisfying all but k of the given constraints, for an abstract class of optimization problems introduced by Sharir and Welzl—the so-called LP-type problems. We give a general algorithm and discuss its efficient implementations for specific geometric problems. For instance, for the problem of computing the smallest circle enclosing all but k of the given n points in the plane, we obtain an $O(n \log n + k^3 n^\varepsilon)$ algorithm; this improves previous results for k small compared with n but moderately growing. We also establish some results concerning general properties of LP-type problems.

1. Introduction

Smallest Enclosing Circles. We begin by discussion the following geometric problem:

> Given a set P of n points in the plane and an integer q, find the smallest circle enclosing at least q points of P.

This has recently been investigated in several papers [3], [18], [12], [16], [25] and it also motivated this paper. In these previous works, algorithms were obtained with

* This research was supported in part by Charles University Grant No. 351 and Czech Republic Grant GAČR 201/93/2167. Part of this research was performed while the author was visiting the Computer Science Institute, Free University Berlin, and it was supported by the German–Israeli Foundation of Scientific Research and Development (G.I.F.), and part while visiting the Max-Planck Institute for Computer Science in Saarbrücken.

roughly $O(nq)$ running time.[1] There seems to be little hope at present of improving these bounds substantially in the full range of values of q. In particular, as observed by D. Eppstein (private communication), for q close to $n/2$, a subquadratic solution would imply a subquadratic solution[2] for several other basic problems, for which one seems unlikely with present methods (information about this class of problems, collected and publicized under the name "n^2-hard problems" by M. Overmars and his colleagues, can be found in [20]).

In this paper we investigate improvements over the roughly $O(nq)$ bound in the situation when $k = n - q$ is small compared with n (all but few points should be enclosed by the circle); this question was raised, e.g., by Efrat *et al.* [16]. One of the first methods coming to mind for solving the problem is to construct the qth-order Voronoi diagram for the point set P, and find the required circle by inspecting all its cells (this approach was pointed out by Aggarwal *et al.* [3]). It is known that the combinatorial complexity of the qth-order Voronoi diagram is $\Theta((n-q)q)$, and it has been shown recently that it can be constructed in expected time $O(n \log^3 n + (n-q)q \log n)$ [1] (see also [7] and [2] for previous results). In our setting, this says that the smallest circle enclosing all but k points can be found in $O(n \log^3 n + nk \log n)$ time.

In this paper we show that still better can be done for small k, namely, that the problem can be solved in close to linear time with k as large as $n^{1/3}$:

Theorem 1.1. *The smallest circle containing all but at most k of the given n points in the plane can be computed in $O(n \log n + k^3 n^{\varepsilon})$ time.*[3]

A predecessor of our technique is a result of Megiddo [27], who demonstrated that the $k = 0$ case, the smallest enclosing circle for n points, can be solved in $O(n)$ time.

LP-Type Problems. The problem of finding the smallest enclosing circle belongs to a class of optimization problems known as LP-type problems (or "generalized linear programming" problems). This class was introduced by Sharir and Welzl [32]. It captures the properties of linear programming relevant for the description and analysis of their linear programming algorithm. The definition and more information on LP-type problems is given later.

Each LP-type problem has an associated parameter, the so-called combinatorial dimension (or dimension for short). For instance, for a feasible linear programming

[1] Here are the specific running times: Eppstein and Erickson [18] solve the problem in $O(nq \log q + n \log n)$ time with $O(n \log n + nq + q^2 \log q)$ space, and Datta *et al.* [12] give an algorithm with the same running time and space improved to $O(n + q^2 \log q)$. Efrat *et al.* [16] achieve $O(nq \log^2 n)$ time with $O(nq)$ space or alternatively $O(nq \log^2 n \log(n/q))$ time with $O(n \log n)$ space, and the author [5] has $O(n \log n + nq)$ time with $O(nq)$ space or time $O(n \log n + nq \log q)$ with $O(n)$ space.

[2] Here "subquadratic" means $O(n^{2-\delta})$ for a constant $\delta > 0$.

[3] Throughout this paper, ε in exponents stands for a positive constant which can be made arbitrarily small by adjusting the parameters of the algorithms. Multiplicative constants implicit in the $O(\ )$ notation may depend on ε.

problem (one with a solution satisfying all the constraints), this combinatorial dimension equals the geometric dimension of the underlying space. Randomized algorithms are known for solving an LP-type problem with n constraints and of dimension bounded by a constant in $O(n)$ expected time, see [8] and [32] (also [6] for a deterministic algorithm and [23] and [26] for algorithms with expected running time subexponential in the dimension). The algorithm have mostly been developed for linear programming, but under suitable computational assumptions they work for all LP-type problems, which includes many other optimization problems of geometric flavor, such as the smallest enclosing circle computation or finding the distance of two convex polyhedra. See also [21] and [4].

The problem of finding the best solution satisfying all but at most k of the given constraints can be posed and solved within the framework of LP-type problems. We need the considered LP-type problem to be *nondegenerate*. In geometric situations this roughly means that the constraints are in general position, which can usually be achieved by perturbation techniques similar to the ones for removing degeneracies in geometric problems. However, one has to proceed carefully so that the combinatorial structure of the problem is not changed in an undesirable way. Exact definitions for the notions in the following theorem are given in the next section.

Theorem 1.2. *Let d be a constant, let (H, w) be a nondegenerate LP-type problem of dimension d with n constraints, and let k be a given integer, $1 \leq k < n$. Then the minimum basis violated by at most k of the constraints can be computed in $O(nk^d)$ time, by finding optima for $O(k^d)$ LP-type problems of the form (G, w) with $G \subseteq H$.*

Paper Overview and Further Results. In Section 2, we first review the definitions and some properties for LP-type problems (Section 2.1). Then we define and discuss nondegenerate LP-type problems (Section 2.2). In Section 2.3 we consider bounds on the number of bases violated by at most k constraints and by exactly k constraints, phrasing results and proofs known for linear programming in the LP-type problems setting. These sections also include facts, observations, and questions which are not directly relevant for the considered algorithms at present, but we consider them of some independent interest. In Section 2.4 we prove Theorem 1.2.

The general algorithm can sometimes be implemented more efficiently in specific geometric situations using dynamic data structures. Theorem 1.1 is one such case. In Section 3 we discuss this plus the following problem (derived from linear programming):

Problem 1.3. Given a collection H of n closed half-spaces in $\mathbb{R}^d$ and an integer k, $1 \leq k < n$, find the lexicographically smallest point of $\mathbb{R}^d$ contained in all but at most k of the half-spaces of H.

Other geometric problems amenable to a similar treatment are only mentioned without going into details.

After we deal with the technical obstructions concerning degeneracies, Problem 1.3 can be solved using the general algorithm from Theorem 1.2. There are two substantially different cases: if the linear programming problem defined by H is

feasible (that is, the intersection of all half-spaces of H is nonempty), then the dimension of the corresponding LP-type problem is d, while for an infeasible problem the dimension is $d+1$ (note that "feasible," "infeasible" refers to the existence of a point in all half-spaces of H, not to the existence of a point lying in all but at most k half-spaces of H).

By using dynamic data structures, the complexity of the general algorithm can be improved. The results are summarized in the following theorem (we do not give the improved complexities for larger dimensions, as the improvements become less significant):

Theorem 1.4.

(i) *In the* feasible case *Problem* 1.3 *can be solved*:
- *in time* $O(nk^d)$ *by the general algorithm, for any fixed* d,
- *in time* $O(n \log n + k^3 n^\varepsilon)$ *in dimension* 3, *and*
- *in time* $O(n \log n + k^{8/3} n^{2/3+\varepsilon} + k^4 n^{1/3+\varepsilon})$ *in dimension* 4.

(ii) *In the* infeasible case *we get*:
- $O(nk^{d+1})$ *time for the general algorithm,*
- $O(n \log n + k^3 \log^2 n)$ *time in dimension* 2, *and*
- $O(n \log n + k^4 n^\varepsilon)$ *time in dimension* 3.

We summarize previous work on Problem 1.3. In the *feasible case* the problem can be rephrased as finding the lexicographically smallest point of the k-level in an arrangement of n hyperplanes. An early paper considering the problem is [22]. In the plane there is an $O(n \log^2 n)$ algorithm (this is why we do not mention the planar case in Theorem 1.4). It is based on the observation that, for a given horizontal line, it can be determined in $O(n \log n)$ time whether it intersects the k-level. Then parametric search is used to determine the lowest line which still intersects the k-level. This algorithm (implicitly) appears in [11]. It is not clear at present how efficiently this approach can be generalized to higher dimensions; a roughly $O(n^{d-1})$ algorithm seems straightforward.

A natural approach in higher dimensions is to construct the k-level or the $(\leq k)$-level in the arrangement.[4] In dimension 3, an output-sensitive algorithm is known which constructs the k-level in $O(n^\varepsilon(b+n))$ time, where b denotes the complexity of the constructed level [2]. The worst-case complexity of the k-level is $O(n^{8/3})$ [13] (see also [5]). In higher dimensions only weak worst-case bounds on the k-level complexity are known, and also output-sensitive construction algorithms become less effective, so that constructing the whole $(\leq k)$-level may be considered. This can be done in expected $O(nk^2 \log(n/k))$ time in dimension 3 and in worst-case optimal $O(n^{\lfloor d/2 \rfloor} k^{\lceil d/2 \rceil})$ expected time in dimension $d \geq 4$ [28]. For k much smaller than n, the level construction approach thus becomes quite inefficient in higher dimensions. We remark that the picture is very different for "random" problem

[4] Even in the plane, the above mentioned $O(n \log^2 n)$ algorithm can be improved a little for k very small, see [19].

instances, as Mulmuley [28] proves that, for hyperplanes randomly chosen from several natural distributions, the $(\leq k)$-level expected complexity as well as expected construction time are $O(nk^{d-1})$.

Problem 1.3 in the *infeasible case* was investigated in several papers. For instance, Cole *et al.* [11] and Everett *et al.* [19] considered the following *weak separation* problem in the plane: given two point sets R and B with n points in total, find the smallest k for which there is a line l and a k-point set $E \subseteq R \cup B$ such that all points of $R \setminus E$ lie on one side of l and all points of $B \setminus E$ lie on the other side. In a dual setting, this amounts to finding the smallest k for which there is a point in the plane contained in all but k of the given half-planes. Everett *et al.* [19] provide an $O(nk \log k + n \log n)$ solution, based on a $(\leq k)$-level construction algorithm. Efrat *et al.* [15] consider the problem in the latter form and obtain similar results. Moreover, they mention some more applications and investigate the analogous problem in dimension 3, where they give an $O(nk^2(\log n + \log^2(n/k))$ solution. Another closely related problem is the finding of a line intersecting all but at most k of given n vertical segments in the plane, investigated by Everett *et al.* [19]; it can also be reduced to linear programming with at most k violated constraints.

Our results for both the two- and three-dimensional problem thus present an improvement over the previous roughly $O(nk)$ bounds if k is in the range (roughly) from $\log n$ to $\sqrt{n}$. We remark that the smallest k (in the weak separation problem or its dual) can be found in the same time as are the bounds in Theorem 1.4 (where the k is given), since our method searches all bases of level at most k, and we may arrange it so that bases of smaller level are searched first.

2. LP-Type Problems

2.1. Basic Definitions and Properties

We begin by recalling the abstract framework of [32] (with minor formal modifications). A *minimization problem* is a pair (H, w), where H is a finite set, and $w: 2^H \to \mathscr{W}$ is a function with values in a linearly ordered set $(\mathscr{W}, \leq)$. The elements of H are called the *constraints*, and, for a subset $G \subseteq H$, $w(G)$ is called the *value* of G.

Intuitively, the value $w(G)$ for a subset G of constraints stands for the smallest value attainable for a certain objective function while satisfying all the constraints of G. The goal is to find $w(H)$. For the computation, the problem is not specified by giving the value of w for each subset (which would make the computation trivial), but rather by oracles implementing certain primitive operations, to be described below.

The set $\mathscr{W}$ is assumed to possess a smallest element denoted by $-\infty$ (standing for "optimum undefined") and usually also a largest element ∞ (with intuitive meaning "no feasible solution exists").

The minimization problem (H, w) is called an *LP-type problem* if the following two axioms are satisfied:

Axiom 1 (Monotonicity). *For any F, G with $F \subseteq G \subseteq H$, $w(F) \leq w(G)$.*

Axiom 2 (Locality). *For any $F \subseteq G \subseteq H$ with $w(F) = w(G) > -\infty$ and any $h \in H$, $w(G \cup \{h\}) > w(G)$ implies that also $w(F \cup \{h\}) > w(F)$.*

Before we specify the computational primitives, we introduce some more terminology. A *basis* B is a set of constraints with $w(B') < w(B)$ for all proper subsets B' of B. A *basis for* a subset G of H is a basis $B \subseteq G$ with $w(B) = w(G)$. So a basis for G is a minimal subset of G with the same value as G. The maximum cardinality of any basis is called the *dimension of* (H, w) and is denoted by $\dim(H, w)$.

We say that a constraint $h \in H$ *violates* a set G if $w(G \cup \{h\}) > w(G)$. This notion is most often used with G being a basis. For $G \subseteq H$ we denote by $V(G)$ the set of constraints of H violating G.

As was mentioned in the Introduction, there are several algorithms for solving an LP-type problem of constant bounded dimension in time $O(|H|)$. These algorithms differ slightly in the assumptions on the primitive operations available for the considered problem. We describe the primitives needed for the randomized algorithm of Sharir and Welzl [32].

Violation Test. Given a basis B and a constraint $h \in H$, decide whether h violates B.

Basis Change. Given a basis B and a constraint h, return (some) basis for $B \cup \{h\}$.

Initial Basis. At the beginning, we have some basis B_0 with $w(B_0) > -\infty$.

For the $O(nk^d)$ time bound in Theorem 1.2, we assume that both violation test and basis change are implemented in constant time, and that for any $G \subseteq H$ with $w(G) > -\infty$ an initial basis $B_0 \subseteq G$ with $w(B_0) > -\infty$ can be found in $O(n)$ time.

To illustrate the definitions, we specify how two particular problems fit into this framework (for more details see [26]). For the problem of finding the *smallest enclosing circle* for a set H of points in the plane, the constraints are the points, the value $w(G)$ of a nonempty set $G \subseteq H$ is the radius of the smallest circle enclosing G, and we set $w(\emptyset) = -\infty$; thus the set $\mathscr{W}$ consists of the element $-\infty$ plus the nonnegative real numbers. Bases have cardinality 0, 1, 2, or 3. Implementing the computational primitives is straightforward.

The *linear programming problem* in $\mathbb{R}^d$ is considered in the following form: We are given a set H of closed half-spaces in $\mathbb{R}^d$, and the goal is to find the point x in the intersection of the half-spaces of H with the lexicographically smallest coordinate vector. To overcome various technicalities, we implicitly add the constraints "$x_i \geq -K$" to the problem, for $i = 1, 2, \ldots, d$ and with K standing for a very large number. The set $\mathscr{W}$ is thus $[-K, \infty)^d$ ordered lexicographically plus a largest element ∞, and the value $w(G)$ of a set of constraints is defined as the lexicographically smallest point (vertex) of the intersection of all half-spaces of G in $[-K, \infty)^d$ or ∞ if no such point exists. A basis for a feasible set of constraints has at most d elements, an infeasible basis may have $d + 1$ elements (as $d + 1$ half-spaces in general position are needed to witness the infeasibility). Violation tests and basis changes are implemented easily using Gauss elimination, and we may choose the empty set as an initial basis in our setting.

2.2. *Nondegenerate LP-Type Problems*

The linear programming algorithms of Clarkson [8] and of Sharir and Welzl [32] do not require any special treatment of degenerate input configurations (such as many half-space boundaries passing through a single point). Probably for this reason, no notion of "general position" in the context of LP-type problems has been developed. For our algorithm below, some "nondegeneracy" is important, so we suggest a definition and make few observations concerning nondegenerate LP-type problems.

Definition 2.1. An LP-type problem (H, w) is *nondegenerate* if $w(B) \neq w(B')$ for any two distinct bases B, B'.

An LP-type problem $(H, \overline{w})$ is a *refinement* of an LP-type problem (H, w) if, for any $G, G' \subseteq H$, $w(G) < w(G')$ implies $\overline{w}(G) < \overline{w}(G')$.

For a nondegenerate problem, we write $B(G)$ for the (unique) basis for a set G.

By "removing degeneracies" for an LP-type problem we mean finding some its nondegenerate refinement. Clearly, if B is a basis in (H, w), it is also a basis in $(H, \overline{w})$, but not conversely.

When considering an LP-type problem (H, w) and its refinement $(H, \overline{w})$, we let the notation $V(G)$ (resp. $B(G)$) refer to the set of violating constraints for G (resp. the basis for G in (H, w)), and we use $\overline{V}(G)$ ($\overline{B}(G)$) for violating constraints and basis in $(H, \overline{w})$.

There are two different types of degeneracies. To elucidate the difference, call two bases B, B' *equivalent* if they have identical sets of violating constraints, $V(B) = V(B')$. It is fairly easy to find a refinement of any LP-type problem where no two nonequivalent bases have the same value, as follows.

Define a linearly ordered set $\overline{\mathscr{W}} = \{-\infty) \cup ((\mathscr{W} \setminus \{-\infty\}) \times 2^H)$, where $-\infty$ remains the smallest element, the pairs in the Cartesian product are ordered lexicographically, and the ordering in the second coordinate is some arbitrary linear ordering of the set of all subsets of H. We define a new value function $\overline{w}$ by setting $\overline{w}(G) = (w(G), V(G))$; it is easy to check that this yields a refinement where no two nonequivalent bases have the same value.[5]

Two equivalent bases must always have the same value. Any two equivalent but distinct bases thus violate the nondegeneracy condition, and this is an "intrinsic" degeneracy. For instance, for linear programming, such degeneracies correspond to several bases defining the same vertex.

At present we are not aware of any universal and efficient method for removing this type of degeneracy. The following simple example shows that sometimes any nondegenerate refinement must have a larger dimension than the original (degenerate) problem.

Example 2.2. Let a, b, c, d be the vertices of a square in the plane in clockwise order, and for $G \subseteq H = \{a, b, c, d\}$ define the value $w(G)$ as the circumradius of G.

[5] This modification of the weight function is not reflected in the computation of the algorithms which use only the primitive operations mentioned above, as none of these primitives are changed.

This is an LP-type problem of dimension 2, as any nontrivial circumscribed circle is defined by a diametrical pair. The sets $\{a, c\}$ and $\{b, d\}$ are two equivalent bases. Suppose there is a nondegenerate refinement $(H, \overline{w})$ of this problem with no basis having more than two points. Then without loss of generality we may assume that $\overline{w}(\{a, c\}) < \overline{w}(\{b, d\})$. Set $G = \{a, b, c\}$, $F = \{a, c\}$, and $h = d$; then $\overline{w}(F) = \overline{w}(F \cup \{h\}) = \overline{w}(G) < \overline{w}(G \cup \{h\})$, which violates the locality axiom. Thus, any nondegenerate refinement has dimension at least 3.

For specific geometric problems, it can often be assumed that the corresponding geometric configurations are nondegenerate in a suitable sense (e.g., for linear programming, the boundary hyperplanes of the constraints are in general position and not parallel to any coordinate axis), using the techniques of *simulation of simplicity* for geometric problems, see, e.g., [14], [33], [17], and [31]. However, for instance, this is not enough for an infeasible linear programming problem. There may be many bases witnessing the infeasibility, and by the treatment of linear programming indicated above, all these bases receive the same value ∞. Another (related) serious problem is that if the feasible region of a degenerate linear programming problem consists of a single point (say), then some perturbations of the constraints may make the problem infeasible. Obtaining nondegenerate refinements in such cases may be a somewhat subtle matter; we discuss some possible approaches in Section 3.

2.3. *Bases of Level k*

Let (H, w) be an LP-type problem and let $B \subseteq H$ be a basis. We define the *level* of B as $|V(B)|$, i.e., the number of constraints violating B.

We denote by $\mathscr{B}_k$ the set of all bases of level k, and by $\mathscr{B}_k^{(i)}$ the bases of level k and cardinality i. We use the notation $\mathscr{B}_{\leq k}$ for the set of bases of level at most k.

In this section we discuss bounds for the maximum possible size of $\mathscr{B}_{\leq k}$ and $\mathscr{B}_k$. In the context of linear programming (more exactly, for a feasible linear program), a basis of level k translates to a local minimum of the k-level in the arrangement of hyperplanes bounding the constraints. Mulmuley [28] showed that the total number of such local minima of levels $1, 2, \ldots, k$ in dimension d is $O(k^d)$. His proof is a straightforward adaptation of the method of Clarkson and Shor [10]. Clarkson [9] and independently Mulmuley [29] proved that the number of local minima for level k alone is $O(k^{d-1})$. Both bounds are easily seen to be tight.

In the following theorem we generalize these results for *nondegenerate* LP-type problems (the nondegeneracy is crucial; without it the bounds in the theorem below need not hold). The proof for level at most k goes through unchanged. For the exactly k level bound, we need an extra assumption, namely, that no value $-\infty$ appears in the considered problem. Then we can imitate Clarkson's proof quite closely. We do not know whether the statement remains true without this assumption. Part (ii) is not needed in subsequent developments.

Theorem 2.3.

(i) *For any nondegenerate LP-type problem of dimension d, the number of bases of level at most k and of cardinality at most i does not exceed $e(k+1)^i$. In particular, $|\mathscr{B}_{\leq k}| = O((k+1)^d)$.*

(ii) *For any nondegenerate LP-type problem (H, w) of dimension d with $w(G) > -\infty$ for any $G \subseteq H$, we have $|\mathscr{B}_k| = O((k+1)^{d-1})$.*

Proof. (i) As remarked above, the proof is identical to Mulmuley's proof for the linear programming setting; we recall it for the reader's convenience. For $k = 0$ the bound holds by nondegeneracy, so let $k \geq 1$. Let H be the set of constraints, $n = |H|$, and fix a probability $p = 1/(k+1)$. Draw a sample $S \subseteq H$ by independently picking each constraint of H into S with probability p. A given basis B is a basis for S iff $B \subseteq S$ and $S \cap V(B) = \emptyset$. Thus, the probability of B becoming a basis for S is $p^{|B|}(1-p)^{|V(B)|}$, and if there are N bases of level $\leq k$ and cardinality $\leq i$, the expected number of these bases which become a basis for S is at least $Np^i(1-p)^k$. On the other hand, S has only one basis by the nondegeneracy assumption, from which we get $N \leq p^{-i}(1-p)^{-k} \leq e(k+1)^i$.

(ii) Here we follow Clarkson's proof. From part (i), we know that the number of bases of level k of cardinality $< d$ is $O(k^{d-1})$. It remains to bound the number of bases of level k of cardinality exactly d.

First, we assume that for each $G \subseteq H$ with $|G| \geq d$ the basis for G has cardinality exactly d (such LP-type problems are called *basis-regular* in [26]). Let i be an integer, $d \leq i \leq n = |H|$, and choose a random i-tuple $S \subseteq H$. The basis for S is unique and it belongs to $\mathscr{B}_k^{(d)}$ for some k. For a fixed basis $B \in \mathscr{B}_k^{(d)}$, the probability of B becoming the basis for S is

$$\binom{n-d-k}{i-d} \Big/ \binom{n}{i}.$$

Hence both sides of the following equality express the expected number of bases for S:

$$1 = \sum_{k=1}^{n-d} b_k \frac{\binom{n-d-k}{i-d}}{\binom{n}{i}},$$

where $b_k = |\mathscr{B}_k^{(d)}|$. It can be checked that the only system of numbers $b_0, b_1, \ldots, b_{n-d}$ satisfying all these equations for $i = d, d+1, \ldots, n$ is given by

$$b_k = \binom{k+d-1}{d-1}.$$

So far we proved (ii) for a basis-regular problem. The proof is finished by establishing the following lemma:

Lemma 2.4. *Let (H, w) be a nondegenerate LP-type problem of dimension d, such that $w(G) > -\infty$ for any $G \subseteq H$. Then a nondegenerate, d-dimensional refinement $(H, \overline{w})$ of (H, w) exists such that any $G \subseteq H$ with $|G| \geq d$ has a basis of cardinality exactly d in $(H, \overline{w})$ and, for any d-element basis B in (H, w) (which is necessarily also a basis in $(H, \overline{w})$), we have $V(B) = \overline{V}(B)$.*

Proof. We fix an arbitrary linear ordering on H. Let $[H]^{\leq d}$ denote the set of all subsets of H of cardinality at most d. We define a linear ordering on $[H]^{\leq d}$ by first ordering the subsets by the number of elements (smaller sets coming first), and ordering the sets of each size lexicographically: first we compare the largest elements, then if they coincide we compare the second largest elements, etc.

Let $\mathscr{W}$ be the range of the mapping w. We define $\overline{\mathscr{W}}$, the range of the mapping $\overline{w}$, as the Cartesian product $\mathscr{W} \times [H]^{\leq d}$, ordered lexicographically. Finally the mapping $\overline{w}\colon 2^H \to \overline{\mathscr{W}}$ is defined as follows: For a set $G \subseteq H$, the first component of $\overline{w}(G)$ is $w(G)$. To determine the second component, we set $m = m(G) = \min(d, |G|) - |B(G)|$, and we let the second component of $\overline{w}(G)$ be the set consisting of the m largest elements of the set $G \setminus B(G)$. This finishes the definition of $\overline{w}$ and it remains to verify the required properties, which is routine.

Clearly, $w(G) < w(G')$ implies $\overline{w}(G) < \overline{w}(G')$, for any G, G'.

It is easy to see that, for $G \subseteq H$, G has a unique basis $\overline{B}(G)$ (recall that $\overline{B}(G)$ refers to the basis in $(H, \overline{w})$), which has the form $B(G) \cup \{g_1, \ldots, g_m\}$, where $m = m(G)$ and $g_1, \ldots, g_m$ are the m largest elements of $G \setminus B(G)$. Any such basis $\overline{B}$ can be reconstructed from its value $\overline{w}(\overline{B})$, which shows that the problem is nondegenerate. We have $|\overline{B}(G)| = |B(G)| + m(G) = \min(d, |G|)$, so $\dim(H, \overline{w}) = d$.

We verify the two axioms for LP-type problems for $(H, \overline{w})$. For monotonicity, we consider sets $F \subset G$. If $w(F) < w(G)$, then also $\overline{w}(F) < \overline{w}(G)$, so let $w(F) = w(G)$. Then also $B(F) = B(G)$, and so $F \setminus B(F) \subset G \setminus B(G)$ and $m(G) \geq m(F)$. Thus, the second component of $\overline{w}(F)$ does not exceed the second component of $\overline{w}(G)$ and $\overline{w}(F) \leq \overline{w}(G)$ as desired.

To check locality, consider $F \subset G \subset G \cup \{h\}$ with $\overline{w}(F) = \overline{w}(G) < \overline{w}(G \cup \{h\})$. Both F and G have the same basis $\overline{B}$ in $(H, \overline{w})$, which has the form $\overline{B} = B \cup \{g_1, \ldots, g_m\}$, where $B = B(F) = B(G)$ and $g_1, \ldots, g_m$ are the m largest elements in both $G \setminus B$ and $F \setminus B$. In particular, it must be $m(F) = m(G)$, and this is possible only if $|F| \geqq d$.

We distinguish two cases. If $w(G \cup \{h\}) > w(G)$, then by locality in (H, w) (and since $w(F) > -\infty$ by our assumption) we get $w(F \cup \{h\}) > w(F)$ and hence also $\overline{w}(F \cup \{h\}) > \overline{w}(F)$. Assume now $w(G \cup \{h\}) = w(G)$; this means that $B(G \cup \{h\}) = B(G) = B(F) = B$. Since $B(G \cup \{h\}) = B(F)$ and $|F| \geq d$, we get $m(G \cup \{h\}) = m(F) = m(G)$. Then $\overline{w}(G \cup \{h\}) > \overline{w}(G)$ means that h occurs among the m largest elements of $G \cup \{h\} \setminus B$, and hence also among the m largest elements of $F \cup \{h\} \setminus B$, which in turn gives $\overline{w}(F \cup \{h\}) > \overline{w}(F)$.

Finally let B be a d-element basis in (H, w). If $w(B \cup \{h\}) = w(B)$, then $B(B \cup \{h\}) = B$ and $\overline{w}(B \cup \{h\}) = (B, \emptyset) = \overline{w}(B)$, so a constraint violates B in $(H, \overline{w})$ iff it violates it in (H, w), which concludes the proof. $\square$

2.4. *Finding the Minimum k-Level Basis*

Nondegeneracy. In our algorithm we need to assume that we deal with a nondegenerate LP-type problem. The following easy proposition shows that in order to find a minimal $(\leq k)$-level basis in a possibly degenerate LP-type problem (H, w), it suffices to find the minimum $(\leq k)$-level basis in any nondegenerate refinement of (H, w).

Proposition 2.5. *Let (H, w) be an LP-type problem, let $(H, \overline{w})$ be its nondegenerate refinement. Let $\overline{B}$ be the basis in $\overline{\mathscr{B}}_{\leq k}$ (the set of all bases of level at most k in $(H, \overline{w})$) with the minimum $\overline{w}$-value, let B be a basis for $\overline{B}$ in (H, w). Then $B \in \mathscr{B}_{\leq k}$ and $w(B)$ is minimum over $\mathscr{B}_{\leq k}$.*

Proof. We have $V(B) = V(\overline{B}) \subseteq \overline{V}(\overline{B})$, so $B \in \mathscr{B}_{\leq k}$. For contradiction, suppose there is a basis $B_1 \in \mathscr{B}_{\leq k}$ with $w(B_1) < w(B)$. Put $G_1 = H \setminus V(B_1)$ and $\overline{B}_1 = \overline{B}(G_1)$. Any constraint violating $\overline{B}_1$ in $(H, \overline{w})$ must be outside G_1, so $\overline{V}(\overline{B}_1) \subseteq V(B_1)$ and $B_1 \in \overline{\mathscr{B}}_{\leq k}$. Since $\overline{w}(\overline{B}_1) = \overline{w}(G_1)$, we have $w(\overline{B}_1) = w(G_1) = w(B_1) < w(B) = w(\overline{B})$ (the first and last equalities follow from the refinement condition) and hence $\overline{w}(\overline{B}_1) < \overline{w}(\overline{B})$, a contradiction with the choice of $\overline{B}$. □

The Algorithm. For the rest of this section, let (H, w) be a nondegenerate LP-type problem of dimension d, and let $k \leq n$ be a given integer. Our goal is to find the basis with the smallest value among all bases of level at most k in (H, w). The algorithm consists of searching all bases of level at most k and selecting the one with minimum value. It is easy to see that, for a nondegenerate problem, the optimal basis has level exactly k, but our method requires also searching bases of smaller levels.

First we define a directed acyclic graph on the vertex set $\mathscr{B}_{\leq k}$. An edge goes from a basis $B \in \mathscr{B}_j$ to a basis $B' \in \mathscr{B}_{j+1}$ $(0 \leq j < k)$ if $V(B') = V(B) \cup \{b\}$ for some $b \in B$.

We note that B' is the basis for $H \setminus V(B) \setminus \{b\}$. Therefore, given $B \in \mathscr{B}_j$, we can find all its neighbors in $\mathscr{B}_{j+1}$ by computing, for every $b \in B$, the basis B' for $H \setminus V(B) \setminus \{b\}$ and checking the condition $V(B') = V(B) \cup \{b\}$. This requires solving $|B| \leq d$ LP-type problems of the form (G, w) with $G \subseteq H$ plus simple auxiliary operations.

We need the following:

Lemma 2.6. *Every basis of $\mathscr{B}_k$ can be reached from $B(H)$ by a directed path.*

Proof. It suffices to show that any basis $B' \in \mathscr{B}_{j+1}$ has a predecessor in $\mathscr{B}_j$.

Write $G = H \setminus V(B')$, then $B' = B(G)$. For every $h \in V(B') \neq \emptyset$, consider the value $w(G \cup \{h\})$, and let $h_0 \in V(B')$ be an element giving the smallest of these values. In fact, such h_0 is unique, as $w(G \cup \{h_0\}) = w(G \cup \{h\})$ for $h \neq h_0$ implies that $B(G \cup \{h_0\}) = B(G \cup \{h\}) \subseteq G$ by nondegeneracy, and this in turn means $w(G \cup \{h_0\}) = w(G)$, a contradiction with the assumption $h_0 \in V(B')$.

Let B be the basis for $G \cup \{h_0\}$. We claim that B is the desired predecessor of B'. We have $V(B) \cup \{h_0\} \subseteq V(B')$; we need to show equality. Suppose the inclusion is proper, then some $h \in V(B')$, $h \neq h_0$, does not violate B. Then, by locality, h does not violate $G \cup \{h_0\}$ either, so $w(G \cup \{h_0\}) = w(G \cup \{h, h_0\})$, but we have $w(G \cup \{h\}) > w(G \cup \{h_0\})$ by the choice of h_0, a contradiction with monotonicity. □

We are ready to finish the proof of Theorem 1.2. We start in the basis $B(H)$, and we search the above-defined directed acyclic graph by some graph traversal algorithm, say by depth-first traversal. For any current node, we can find all of its at most d successors in $O(n)$ time, and by Theorem 2.3(i), we know that only $O(k^d)$ bases need to be searched. By the above lemma, we know that all bases of $\mathscr{B}_{\leq k}$ are reached. □

A Scheme for Applying Suitable Dynamic Data Structures. When traversing the graph defined above, we can maintain two dynamic data structures, $\mathscr{O}$ and $\mathscr{V}$. For a current basis B, the data structure $\mathscr{O}$ stores the constraints of $H \setminus V(B)$, and it can determine the basis for the currently stored set of constraints. The data structure $\mathscr{V}$ stores the constraints of $V(B)$, and it can test whether all constraints of the currently stored set violate a basis given as a query.

For testing if a successor basis B' of B with $V(B') = V(B) \cup \{b\}$ for some $b \in B$ exists, we first delete b from $\mathscr{O}$ and then we query $\mathscr{O}$ for the basis B' for the current constraint set. Then we query the data structure $\mathscr{V}$ to check whether $V(B) \subseteq V(B')$. If yes, we insert b into $\mathscr{V}$, and we are ready to continue the search with B' as the current basis. When traversing the previously visited edge (B, B') backward, we insert b to $\mathscr{O}$ and delete it from $\mathscr{V}$.

3. Geometric Applications

In this section we mainly consider Problem 1.3 (minimum vertex contained in all but at most k of given n half-spaces in $\mathbb{R}^d$). We treat a linear programming problem as an LP-type problem in the way outlined in Section 2.1. We let H be the set of the constraints (half-spaces), and for $G \subseteq H$ let $\mathscr{F}(G)$ denote the *feasible region* of G, that is, the intersection of all the half-spaces of H with $[-K, \infty)^d$ (recall that K stands for a large enough number; we need not specify a numeric value, rather we may treat it as a formal quantity). For a half-space h, we let ∂h stand for its bounding hyperplane, and for a set G of half-spaces we write ∂G for $\{\partial h;\ h \in G\}$.

3.1. *Removing Degeneracies*

The input problem may be degenerate. For a feasible linear program, degeneracy is caused by degeneracy in the arrangement of ∂H, while for an infeasible problem, any two distinct infeasible bases present a degeneracy, even if the hyperplanes of ∂H are in general position. For our algorithm, we need a nondegenerate refinement.

For a problem with a nonempty and full-dimensional feasible region, a nondegenerate refinement can be produced relatively easily using infinitesimal perturbations, while the situation gets more complicated in other cases. One potential problem is the following: if the feasible region is nonempty but has empty interior, an arbitrarily small perturbation of the input half-spaces may cause the problem to become infeasible, and in general we need not get a refinement.

At this moment, a general remark concerning simulation of simplicity is perhaps appropriate. When applying simulation of simplicity on a geometric problem, we replace it in effect by a different problem (although an "infinitesimally close" one), and an algorithm run on this different problem may sometimes yield a different answer than the correct one for the original problem (e.g., the original linear program is feasible while the perturbed one is not). We can take two points of view: First, we may say that the input numbers for a "real world" problem are inaccurate anyway, and so the answer for the perturbed problem is equally appropriate as the one for the original problem. For example, we cannot really tell whether the feasible region is very thin or empty. From the second point of view, the input is given exactly and its degeneracies really exist, so if we use perturbation, we must be sure to recover the answer to the original problem correctly. By requiring that our algorithm is run on a nondegenerate refinement of the input problem, we are taking the second of the outlined positions (which may not always be appropriate, depending on the application). The first attitude would allow us to take an arbitrary infinitesimal perturbation, and would save us many complications.

The following proposition shows that a nondegenerate refinement can be constructed for a linear programming problem.

Proposition 3.1. *Given a set H of half-spaces in $\mathbb{R}^d$, one can define a nondegenerate LP-type problem $(H, \overline{w})$, which is a refinement of the LP-type problem (H, w) defined above, has dimension d (resp. $d+1$) if $\mathcal{F}(H) \neq \emptyset$ (resp. $\mathcal{F}(H) = \emptyset$), and for which the computational primitives can be implemented in constant time.*

Proof. We produce a sequence of successive refinements (H, w_j), $j = 1, 2, 3$. For their definition, we use small positive numbers $\varepsilon_1 \gg \varepsilon_2 \gg \varepsilon_3 > 0$. First we fix a small enough $\varepsilon_1 > 0$ (depending on H and K, the number appearing in the implicit constraints), then $\varepsilon_2 > 0$ small enough depending moreover on ε_1, and finally ε_3 depending also on ε_2. In the algorithm we treat K, ε_1, ε_2, ε_3 as indeterminates (so we calculate with polynomials), as is usual in methods for simulation of simplicity. The required operations can still be performed in constant time (although a practical implementation would probably be very slow; finding a more practical way for making the problem nondegenerate is a matter for further research).

In the first refinement (H, w_1), we simplify the range of the value function, making it $\mathbb{R} \cup \{\infty\}$. We define a vector $c := (1, \varepsilon_1, \varepsilon_1^2, \ldots, \varepsilon_1^{d-1})$, and we set

$$w_1(G) := \begin{cases} \min\{c \cdot x;\ x \in \mathcal{F}(G)\} & \text{for a feasible } G, \\ \infty & \text{for an infeasible } G. \end{cases}$$

It is easy to see that if we fix $\varepsilon_1 > 0$ small enough, the ordering of the subsets of H by w and by w_1 is exactly the same, so in fact we have isomorphic LP-type problems.

Let H_0 denote the set of half-spaces defined by $x_i \geq -K$, $i = 1, 2, \ldots, d$. We also have the following property for a small enough ε_1:

> The vector c is not perpendicular to any edge of the arrangement of the hyperplanes of $\partial(H \cup H_0)$. (1)

In the second step we produce a perturbed set $H^{(2)}$ of half-spaces, by translating each half-space of H outward by a distance of ε_2, and we define the new value function w_2 by

$$w_2(G) := \begin{cases} \min\{c \cdot x;\ x \in \mathscr{F}(G^{(2)})\} & \text{for a feasible } G^{(2)}, \\ \infty & \text{for an infeasible } G^{(2)}, \end{cases}$$

where $G^{(2)}$ denotes the corresponding perturbation of G. The goal of this perturbation is to assure the following condition:

> for any feasible $G^{(2)} \subseteq H^{(2)}$, the feasible region $\mathscr{F}(G^{(2)})$ has a nonempty interior. (2)

This in turn allows us to use an arbitrary small enough perturbation in the next step. We also note that this step is unnecessary if the original problem already has $\mathscr{F}(H)$ with a nonempty interior.

We check condition (2). For a feasible G, $\mathscr{F}(G^{(2)})$ clearly has a nonempty interior. If G is infeasible, this means that the (open) complements of the half-spaces of $G \cup H_0$ cover $\mathbb{R}^d$. Then some $\varepsilon_2 > 0$ exists such that if we shrink these complements by at most ε_2 they still cover $\mathbb{R}^d$, so $G^{(2)}$ is infeasible as well and (2) holds.

Since the value function is defined by optima in a linear programming problem where each set of constraints defines a unique optimum, (H, w_2) is automatically an LP-type problem of an appropriate dimension (the same will hold for the next refinement, (H, w_3)). It remains to show that (H, w_2) is indeed a refinement of (H, w_1). We already know that feasibility and infeasibility of sets of constraints is preserved. It is also easy to see that if G is feasible, its minimum vertex moves continuously as we translate the half-spaces by a small amount. Therefore, if $w_1(G) < w_1(G')$, we also get $w_2(G) < w_2(G')$ for a sufficiently small ε_2.

In the next step we aim at bringing the hyperplanes of ∂H into general position by perturbing each hyperplane by at most[6] ε_3, where $\varepsilon_3 \ll \varepsilon_2$. We let $H^{(3)}$ denote a perturbed version of $H^{(2)}$, such that any of the hyperplanes of $\partial H^{(3)}$ is perturbed by at most ε_3 with respect to the corresponding hyperplane in $\partial H^{(2)}$. Except for the

[6] The distance of hyperplanes is defined as follows: We consider a hyperplane h given by an equation $a_1x_1 + \cdots + a_dx_d = a_0$, where the coefficients are normalized so that $a_0 \geq 0$ and the Euclidean norm $\|(a_1, \ldots, a_d)\| = 1$. The distance of two hyperplanes is the Euclidean norm of the difference of their coefficient vectors $(a_0, a_1, \ldots, a_d)$.

restriction on size, the perturbation is arbitrary, so we may choose it in such a way that $H^{(3)}$ satisfies all general position requirements we specify later. We define the corresponding value function w_3 using $H^{(3)}$ in the same way as w_2 was defined using $H^{(2)}$.

Since $H^{(3)}$ is in general position, we may assume that any two *feasible* bases in (H, w_3) have distinct values. We want to show that (H, w_3) is a refinement of (H, w_2). To this end, consider sets $G, G' \subseteq H$ with $w_2(G) < w_2(G')$. There are few cases to be distinguished.

First, let both G, G' be feasible with respect to w_2. Put $\delta = (w_2(G') - w_2(G))/4$, and choose a point y in the interior of the feasible region $\mathscr{F}(G^{(2)})$ at distance at most δ from the minimum vertex x of $\mathscr{F}(G^{(2)})$. If the perturbation is small enough, y also remains in $\mathscr{F}(G^{(3)})$, and hence $w_3(G) \le c \cdot y \le c \cdot x + \|c\|\delta \le w_2(G) + 2\delta$.

On the other hand, consider a basis B' (relative to w_2) for G'. The feasible region $\mathscr{F}(B'^{(2)})$ forms a convex cone whose apex x' is its unique minimum (as no face of the cone is perpendicular to c, by (1)). With a small enough continuous movement of the half-spaces defining the cone, the apex moves continuously and remains the c-direction minimum for the cone. We have $\mathscr{F}(G'^{(3)}) \subseteq \mathscr{F}(B'^{(3)})$, so $w_3(G') \ge w_3(B') \ge c \cdot x' - \delta = w_2(G') - \delta$ if the perturbation is small enough, and hence $w_3(G) < w_3(G')$ as desired.

The second case to consider is when $G^{(2)}$ is feasible and $G'^{(2)}$ is infeasible. In the preceding we saw that the value of $w_3(G)$ can be bounded by $w_2(G) + 1$, say, for any sufficiently small ε_3. Hence it suffices to show that $G'^{(3)}$ is either infeasible or $w_3(G'^{(3)})$ becomes arbitrarily large as $\varepsilon_3 \to 0$. Suppose $G'^{(3)}$ is feasible, let x' be the minimum vertex of $\mathscr{F}(G'^{(3)})$ and let $D^{(3)} \subseteq \partial(G'^{(3)} \cup H_0)$ be the d-tuple of hyperplanes defining x'. The corresponding d-tuple $D^{(2)}$ cannot define a vertex (or, rather, it defines a vertex in the infinity), as such a vertex would lie in the complement of some half-space of $G'^{(2)} \cup H_0$ and a small enough perturbation could not make it feasible. From this we can conclude that the norm $\|x'\|$ tends to ∞ with $\varepsilon_3 \to 0$. It remains to show that $c \cdot x'$ is also large. Let $u' = x'/\|x'\|$; we need to show that $c \cdot u'$ does not tend to 0 with $\varepsilon_3 \to 0$. However, for $\varepsilon_3 \to 0$, u' approaches the direction u of an unbounded ray in the arrangement of $D^{(2)} \subseteq \partial(H^{(2)} \cup H_0)$. By condition (1), $c \cdot u$ is bounded away from 0 independently of ε_3, and so, indeed, $c \cdot x' = (c \cdot u')\|x'\| \to \infty$. Hence $w_3(G') > w_3(G)$ and (H, w_3) is a refinement of (H, w_2).

As a last step, we redefine the value function for the infeasible sets, so that a nondegenerate refinement is obtained.

Because of the general position of $H^{(3)}$, we may divide the half-spaces into the upper ones (the ones containing the positive ray of the x_1-axis) and the lower ones. For $G^{(3)} \subseteq H^{(3)}$, let $L(G^{(3)})$ be the set of the lower half-spaces of $G^{(3)}$ and let $U(G^{(3)})$ be the set of the upper half-spaces.

For an infeasible $G^{(3)}$, we define a number $t(G)$ as the minimum amount by which we must shift all the lower half-spaces upward along the x_1-axis so that the feasible region becomes nonempty; formally

$$t(G) := \min\{t; \mathscr{F}((L(G^{(3)}) + (t, 0, \ldots, 0)) \cup U(G^{(3)})) \neq \emptyset\}.$$

We then define the final value function $\overline{w}$:

$$\overline{w}(G) := \begin{cases} w_3(G) & \text{for a feasible } G^{(3)}, \\ (\infty, t(G)) & \text{for an infeasible } G^{(3)}, \end{cases}$$

where the pairs of the form (∞, t) are ordered by the second coordinate and they are larger than the values for feasible sets.

The general position of $H^{(3)}$ allows us to assume that as we shift the lower half-spaces upward, the contact of the feasible regions of the shifted lower half-spaces and of the upper half-spaces occurs at a single point $\tau(G^{(3)})$, and this point is common to at most $d + 1$ boundaries of the half-spaces (some upper ones and some shifted lower ones). These $d + 1$ half-spaces then form the basis for G with respect to $\overline{w}$. A constraint h violates G iff the perturbed half-space $h^{(3)}$ is either an upper one and does not contain the contact point $\tau(G^{(3)})$, or it is a lower one and its translate by $(t, 0, \ldots, 0)$ does not contain $\tau(G^{(3)})$. We leave the checking that $(H, \overline{w})$ is indeed an LP-type problem and a refinement of (H, w_3) to the reader.

We may moreover assume that $t(B) \neq t(B')$ for any two distinct infeasible bases B, B' in (H, w_3), and so $(H, \overline{w})$ is nondegenerate. Finally implementing the computational primitives for $(H, \overline{w})$ is conceptually straightforward for a fixed dimension. □

Remark. Another, perhaps more natural way to order the infeasible bases in an infeasible linear programming problem is to add an extra variable (dimension) to the problem, replacing a constraint $a_1x_1 + \cdots + a_dx_d \leq a_0$ by $a_1x_1 + \cdots + a_dx_d - x_{d+1} \leq a_0$ (which is a well-known trick in linear programming). This makes the problem feasible, and infeasible bases in the original problem correspond to feasible bases with $x_{d+1} > 0$ in the new problem. This does not increase the combinatorial dimension, but dynamic data structures needed for making the algorithm faster would have to work in dimension one higher.

3.2. *Using Dynamic Data Structures*

By proving Proposition 3.1 in the preceding section, we have shown that the general algorithm from Theorem 1.2 is applicable to Problem 1.3. In this section we consider efficient implementation of the dynamic data structures $\mathscr{O}$ and $\mathscr{V}$ mentioned at the end of Section 2.4 for the linear programming problem (actually for its nondegenerate refinement constructed in the proof of Proposition 3.1).

We begin with the feasible case. Here the data structure $\mathscr{V}$ stores a set of half-spaces which all contain a known point (a fixed point in $\mathscr{F}(H)$), and the query boils down to checking whether a given point lies outside of the union of the current set of half-spaces. For this task, algorithms are known with the following performance [2] (we only quote the results relevant to our application): In dimension 3 $O(n^\varepsilon)$ query time can be achieved with $O(n \log n)$ preprocessing time and $O(\log^2 n)$ amortized update time. In dimension $d \geq 4$ the following tradeoff is obtained: for a

parameter m in range $[n, n^{\lfloor d/2 \rfloor}]$, query time $O(n^{1+\varepsilon}/m^{1/\lfloor d/2 \rfloor})$ is obtained with $O(m^{1+\varepsilon})$ space and preprocessing time and with $O(m^{1+\varepsilon}/n)$ amortized update time (in fact, for $m = n$ an $O(n \log n)$ preprocessing suffices). We must apply the appropriate perturbations on the input half-spaces, but this is a simulation of a simplicity technique of a particular type and it only slows down the computation by a constant factor. The same performance can be achieved for the data structure $\mathscr{O}$, which should return a basis for the current set of half-spaces (and we are guaranteed that the half-spaces have a nonempty intersection), see [2] and [24].

In dimension 3 we perform $O(k^3)$ queries and updates in both data structures, which leads to the claimed $O(n \log n + k^3 n^{\varepsilon})$ complexity. For dimension 4 we choose a suitable tradeoff between the total update and query time and the preprocessing time of the data structures, as follows: for $k < n^{1/8}$, we let $m = n$, for $k^4 \leq n < k^8$ we let $m = k^{8/3} n^{2/3}$, and for larger k we choose $m = n^{4/3}$. This yields the formula stated in the theorem. Tradeoffs can also be computed for higher dimensions, although with less significant gains in efficiency. This establishes part (i) of Theorem 1.4.

In the infeasible case the data structure $\mathscr{V}$ stores the upper half-spaces and lower half-spaces separately. Testing for a feasible basis is as before. For an infeasible basis B, we first compute the value $t(B)$ and the contact vertex $\tau = \tau(B^{(3)})$ (see the proof of Proposition 2.5 for definitions), and then we test if τ lies outside all upper half-spaces and $\tau - (t, 0, \ldots, 0)$ lies outside all lower half-spaces.

The data structure $\mathscr{O}$ also stores upper and lower half-spaces separately. If the currently stored set is feasible, the algorithm of [24] returns the basis. If infeasibility is reported, we use parametric search to find the value of t. As a generic algorithm, we use the algorithm of [24] for testing feasibility of the set of the upper half-spaces plus the lower half-spaces shifted by a generic value of t. A more detailed exposition would require explaining the feasibility testing algorithm and we omit it, as the details are easy to fill in, assuming familiarity with [24]. The application of parametric search on top of the feasibility testing algorithm only increases the query time by polylogarithmic factors. The overall performance of the resulting dynamic data structures $\mathscr{V}$ and $\mathscr{O}$ is thus the same as in the feasible case above; this gives the result for dimension 3 in Theorem 1.4(ii).

For the planar case, the parametric search machinery is unnecessary, and we may directly use a relatively simple dynamic data structure for maintaining convex hulls in the plane due to Overmars and van Leeuwen [30]. To build $\mathscr{O}$, we use this data structure in a dual form. One part represents the intersection U of the upper half-planes, another part the intersection L of the lower half-planes. After $O(n \log n)$ preprocessing, half-planes can be inserted and deleted in $O(\log^2 n)$ time, and the data structure provides a representation of the convex chains forming the boundaries of L and U (the chains are stored in balanced binary trees).

If the current problem is feasible, the optimal vertex is either the extreme vertex of U, or the extreme vertex of L, or one of the two intersections of the boundaries of U and of L. All these vertices can be found and examined in $O(\log^2 n)$ time. For an infeasible problem, the first contact of U and L when translating L upward occurs either at a vertex of L or at a vertex of U. For a given vertex v of L, we can determine the point where it hits the boundary of U in $O(\log n)$ time. From the

local situation at that point, we can detect whether the first contact of L and U occurs to the left of v or the right of v. Hence we can determine the first contact, the corresponding t value, and the basis in $O(\log^2 n)$ time by a binary search in the lower convex chain. The data structure $\mathscr{V}$ is also implemented using the Overmars and van Leeuwen data structure, with $O(\log^2 n)$ time per update and $O(\log n)$ time per query. Altogether we get $O(n \log n + k^3 \log^2 n)$ running time. This finishes the proof of Theorem 1.4. □

Proof of Theorem 1.1. For the smallest enclosing circles problem discussed in the Introduction, the situation with a nondegenerate refinement is considerably easier than the one for linear programming (this is because the value function depends continuously on the point set). It suffices to take any sufficiently small perturbation of the input points such that no four points are cocircular and no circle determined by two points passes through another point. The required dynamic data structures are mentioned in [2]. They require $O(n^\varepsilon)$ amortized update time and query time, and this gives the bound in Theorem 1.1. □

There are various other optimization problems of geometric flavor fitting into the LP-type framework, see [26] and [4]. Here are few examples: finding the smallest ball (resp. the smallest volume ellipsoid enclosing a given point set in $\mathbb{R}^d$); finding the largest volume ellipsoid inscribed into the intersection of given half-spaces in $\mathbb{R}^d$; finding the distance of two convex polyhedra given by vertices (resp. by facet hyperplanes in $\mathbb{R}^d$); finding a line transversal for given convex polygons in the plane.

If the dimension is fixed our general algorithm can be applied to the respective derived problems with k violated constraints, provided that the nondegeneracy issue can be handled. In many of the problems, simulation of simplicity alone should suffice; it seems that linear programming is complicated in this respect because feasible solutions need not exist. The applicability of dynamic data structures to speed up the computations must be checked individually. In general, the improvements will probably be significant only if the dimension is really small.

4. Discussion

It would be interesting to find more about nondegenerate refinements of LP-type problems. We have shown that the dimension must sometimes grow at least by one; a natural question is how much growth is necessary and sufficient in the worst case; is there any bound only depending on the dimension?

From a practical point of view, it would be very desirable to have some more direct scheme for making the geometric LP-type problems, linear programming in particular, nondegenerate instead of the rather cumbersome approach via geometric perturbations we used. Alternatively an algorithm might be found which can handle nondegeneracy directly.

In Theorem 2.3(ii) we saw that the number of bases of level exactly k is $O(k^{d-1})$ in a d-dimensional LP-type problem with no $-\infty$ values. Two questions arise naturally: First, is the claim still true if we allow $-\infty$ values? Second, can this result

be used algorithmically for finding the smallest basis of level k, that is, can one efficiently avoid searching all bases of level *at most* k, whose number may be of the order k^d? In particular, for two-dimensional infeasible linear programming, we are actually interested only in two-element bases (as all infeasible bases have the same value ∞ in the original problem), and we know that there are only $O(k^2)$ of these. Still, the current method may search k^3 bases, most of them infeasible ones; could this be avoided?

Acknowledgments

I would like to thank Emo Welzl for bringing the problem to my attention, and Pankaj K. Agarwal and David Eppstein for useful discussions.

References

1. P. Agarwal, M. de Berg, J. Matoušek, and O. Schwarzkopf. Constructing levels in arrangements and higher-order Voronoi diagrams. *Proc. 10th Ann. ACM Symp. on Computational Geometry,* pp. 67–75, 1994.
2. P. K. Agarwal and J. Matoušek. Dynamic half-space range reporting and its applications. *Algorithmica*, 13:325–345, 1995.
3. A. Aggarwal, H. Imai, N. Katoh, and S. Suri. Finding k points with minimum diameter and related problems. *J. Algorithms*, 12:38–56, 1991.
4. N. Amenta. Helly theorems and generalized linear programming. *Discrete Comput. Geom.*, 12:241–261, 1994.
5. B. Aronov, B. Chazelle, H. Edelsbrunner, L. J. Guibas, M. Sharir, and R. Wenger. Points and triangles in the plane and halving planes in space. *Discrete Comput. Geom.*, 6:435–442, 1991.
6. B. Chazelle and J. Matoušek. On linear-time deterministic algorithms for optimization problems in fixed dimension. *Proc. 4th ACM–SIAM Symp. on Discrete Algorithms,* pp. 281–290, 1993.
7. K. L. Clarkson. New applications of random sampling in computational geometry. *Discrete Comput. Geom.*, 2:195–222, 1987.
8. K. L. Clarkson. A Las Vegas algorithm for linear programming when the dimension is small. *Proc. 29th Ann. IEEE Symp. on Foundations of Computer Science,* pp. 452–456, 1988.
9. K. L. Clarkson. A bound on local minima of arrangements that implies the upper bound theorem. Manuscript, 1992.
10. K. L. Clarkson and P. W. Shor. Application of random sampling in computational geometry, II. *Discrete Comput. Geom.*, 4:387–421, 1989.
11. R. Cole, M. Sharir, and C. K. Yap. On k-hulls and related problems. *SIAM J. Comput.*, 16:61–77, 1987.
12. A. Datta, H.-P. Lenhof, C. Schwarz, and M. Smid. Static and dynamic algorithms for k-point clustering problems. *Proc. 3rd Workshop on Algorithms and Data Structures.* Lecture Notes in Computer Science, vol. 709, pp. 265–276. Springer-Verlag, Berlin, 1993.
13. T. Dey and H. Edelsbrunner. Counting triangle crossings and halving planes. *Discrete Comput. Geom.*, 12:281–289, 1994.
14. 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.
15. A. Efrat, M. Lindenbaum, and M. Sharir, Finding maximally consistent sets of halfspaces. *Proc. 5th Canad. Conf. on Computational Geometry,* pp. 432–436, Waterloo, Ontario, 1993.
16. A. Efrat, M. Sharir, and A. Ziv. Computing the smallest k-enclosing circle and related problems. *Proc. 3rd Workshop on Algorithms and Data Structures.* Lecture Notes in Computer Science, vol. 709, pp. 325–336. Springer-Verlag, Berlin, 1993.

17. I. Emiris and J. Canny. An efficient approach to removing geometric degeneracies. *Proc. 8th Ann. ACM Symp. on Computational Geometry,* pp. 74–82, 1992.
18. D. Eppstein and J. Erickson. Iterated nearest neighbors and finding minimal polytopes. *Proc. 4th ACM–SIAM Symp. on Discrete Algorithms,* pp. 64–73, 1993.
19. H. Everett, J.-M. Robert, and M. van Kreveld. An optimal algorithm for the $(\leq k)$-levels, with applications to separation and transversal problems. *Proc. 9th Ann. ACM Symp. on Computational Geometry,* pp. 38–46, 1993.
20. A. Gajentaan and M. H. Overmars. n^2-hard problems in computational geometry. Report RUU-CS-93-15, Department of Computer Science, Utrecht University, Utrecht, April 1993.
21. B. Gärtner. A subexponential algorithm for abstract optimization problems. *Proc. 33rd Ann. IEEE Symp. on Foundations of Computer Science,* pp. 464–472, 1992.
22. D. S. Johnson and F. P. Preparata. The densest hemisphere problem. *Theoret. Comput. Sci.*, 6:93–107, 1978.
23. G. Kalai. A subexponential randomized simplex algorithm. *Proc. 24th Ann. ACM Symp. on Theory of Computing,* pp. 475–482, 1992.
24. J. Matoušek, Linear optimization queries. *J. Algorithms*, 14:432–448, 1993. The results combined with the results of O. Schwarzkopf also appear in *Proc. 8th ACM Symp. on Computational Geometry,* pp. 16–25, 1992.
25. J. Matoušek, On enclosing k points by a circle. *Inform. Process. Lett.*, 53:217–221, 1995.
26. J. Matoušek, M. Sharir, and E. Welzl. A subexponential bound for linear programming. *Proc. 8th Ann. ACM Symp. on Computational Geometry,* pp. 1–8, 1992. Also to appear in *Algorithmica*.
27. N. Megiddo. The weighted Euclidean 1-center problem. *Math. Oper. Res.*, 8(4):498–504, 1983.
28. K. Mulmuley. On levels in arrangements and Voronoi diagrams. *Discrete Comput. Geom.*, 6:307–338, 1991.
29. K. Mulmuley. Dehn–Sommerville relations, upper bound theorem, and levels in arrangements. *Proc. 9th Ann. ACM Symp. on Computational Geometry,* pp. 240–246, 1993.
30. M. H. Overmars and J. van Leeuwen. Maintenance of configurations in the plane. *J. Comput. System Sci.*, 23:166–204, 1981.
31. R. Seidel. The nature and meaning of perturbations in geometric computing. *Proc. 11th Symp. on Theoretical Aspects of Computer Science (STACS).* Lecture Notes in Computer Science. Springer-Verlag, Berlin, 1994.
32. M. Sharir and E. Welzl. A combinatorial bound for linear programming and related problems. *Proc. 1992 Symp. on Theoretical Aspects of Computer Science.* Lecture Notes in Computer Science, vol. 577, pp. 569–579. Springer-Verlag, Berlin, 1992.
33. C. K. Yap. A geometric consistency theorem for a symbolic perturbation scheme. *J. Comput. System Sci.*, 40:2–18, 1990.

Received March 1994, and in revised form July 1994.