APPEARED IN *Algorithmica* **16** (1996) 498-516.

A Subexponential Bound for Linear Programming*†

JIŘÍ MATOUŠEK‡
Department of Applied Mathematics
Charles University, Malostranské nám. 25
118 00 Praha 1, Czechoslovakia
e-mail: matousek@kam.mff.cuni.cz

MICHA SHARIR
School of Mathematical Sciences
Tel Aviv University, Tel Aviv 69978, Israel
e-mail: sharir@math.tau.ac.il
and
Courant Institute of Mathematical Sciences
New York University, New York, NY 10012, USA

EMO WELZL
Institut für Informatik, Freie Universität Berlin
Takustraße 9, 14195 Berlin, Germany
e-mail: emo@inf.fu-berlin.de

Abstract

We present a simple randomized algorithm which solves linear programs with n constraints and d variables in expected

$$\min\{O(d^2 2^d n), e^{2\sqrt{d \ln(n/\sqrt{d}\,)}+O(\sqrt{d}+\ln n)}\}$$

time in the unit cost model (where we count the number of arithmetic operations on the numbers in the input); to be precise, the algorithm computes the lexicographically smallest nonnegative point satisfying n given linear inequalities in d variables. The expectation is over the internal randomizations performed by the algorithm, and holds for any input. In conjunction with Clarkson's linear programming algorithm, this gives an expected bound of

$$O(d^2 n + e^{O(\sqrt{d \ln d})}) \ .$$

The algorithm is presented in an abstract framework, which facilitates its application to several other related problems like computing the smallest enclosing ball (smallest volume enclosing ellipsoid) of n points in d-space, computing the distance of two n-vertex (or n-facet) polytopes in d-space, and others. The subexponential running time can also be established for some of these problems (this relies on some recent results due to Gärtner).

KEYWORDS: computational geometry, combinatorial optimization, linear programming, smallest enclosing ball, smallest enclosing ellipsoid, randomized incremental algorithms.

---

*Work by the first author has been supported by a Humboldt Research Fellowship. Work by the second and third authors has been supported by the German-Israeli Foundation for Scientific Research and Development (G.I.F.). Work by the second author has been supported by Office of Naval Research Grant N00014-90-J-1284, by National Science Foundation Grants CCR-89-01484 and CCR-90-22103, and by grants from the U.S.-Israeli Binational Science Foundation, and the Fund for Basic Research administered by the Israeli Academy of Sciences.

†A preliminary version appeared in *Proc. 8th Annual ACM Symposium on Computational Geometry*, 1992, pages 1–8.

‡Work has been carried out while author was visiting the Institute for Computer Science, Berlin Free University

1 Introduction

Linear programming is one of the basic problems in combinatorial optimization, and as such has received considerable attention in the last four decades. Many algorithms have been proposed for its solution, starting with the simplex algorithm and its relatives [Dan], proceeding through the polynomial-time solutions of Khachiyan [Kha] and Karmarkar [Kar], and continuing with several more recent techniques (reviewed below). While some of the proposed algorithms have proven out to be extremely efficient in practice, analysis of their running time has not been fully satisfactory so far. For example, the simplex algorithm was shown to be exponential in the worst case [KlM]. The algorithms of Khachiyan and of Karmarkar have polynomial bit-complexity, but the number of arithmetic operations they perform depends on the size of the coefficients defining the input and it cannot be bounded solely in terms of n (the number of constraints) and d (the number of variables). In this paper we assume a different model of computation, namely the *real RAM*, widely used in computational geometry. Here the input may contain arbitrary real numbers, and each arithmetic operation with real numbers is charged unit cost. To distinguish complexity bounds in this model from bounds in the bit-complexity model, we call them *combinatorial*.

Until recently, the best known combinatorial bounds were exponential in either n or d; a subexponential (randomized) bound is given in a very recent paper of Kalai [Kal] and also in this paper. One of the major open problems in the area is to find a strongly polynomial algorithm (i.e. of combinatorial polynomial complexity) for linear programming.

In this paper we describe a simple randomized algorithm which solves linear programs with n inequalities and d variables with expected

$$O(\min\{d^2 2^d n, e^{2\sqrt{d\ln(n/\sqrt{d})}+O(\sqrt{d}+\ln n)}\})$$

arithmetic operations. In conjunction with Clarkson's linear programming algorithm [Cla3], this gives an expected bound of

$$O(d^2 n + e^{O(\sqrt{d\ln d})}) \ .$$

The expectation of the running time is with respect to the internal randomizations performed by the algorithm, and holds for any input. This complexity matches that of a recent algorithm due to Kalai [Kal]. (Except for the constant in the exponent, the only significant difference is that Kalai has a version of his algorithm which runs in $e^{O(\sqrt{d})}$ as long as n is linear in d. We can guarantee this bound only for $n = O(\sqrt{d})$.) Chronologically speaking, our algorithm was published first in [ShW], but with a weaker analysis of its running time; Kalai's analysis with a subexponential bound came next, and immediately afterwards we realized that our algorithm also has subexponential running time.

The algorithm is presented in an abstract framework which facilitates the application of the algorithm to a large class of problems, including the computation of smallest enclosing balls (or ellipsoids) of finite point sets in d-space, computing largest balls (ellipsoids) in convex polytopes in d-space, computing the distance between polytopes in d-space, etc. (however, we can guarantee a subexponential running time for only some of these problems; see below for details).

To compare the complexity of our algorithm with other recent techniques, here is a brief survey of some relevant recent literature. Megiddo [Meg2] has given the first deterministic algorithm whose running time is of the form $O(C_d n)$, and is thus linear in n for any fixed d. However the factor C_d in his algorithm is 2^{2^d}; an improvement to $C_d = 3^{d^2}$ can be found in [Dye1] and [Cla1]. Recently, a number of randomized algorithms have been presented for the problem, see [DyF], [Cla3], [Sei], with a better dependence on d, where the best expected running time is given by Clarkson [Cla3]: $O(d^2 n + d^{d/2+O(1)} \log n)$. Nevertheless, the dependence on d is still exponential. The more recent algorithm of Seidel [Sei] has worse expected complexity of $O(d!n)$, but is an extremely simple randomized incremental algorithm. In [Wel] this algorithm was enhanced with a "move-to-front" heuristic, which in practice has drastically improved the performance of the algorithm, but was (and still is) very difficult to analyze. Our algorithm is another variant, in-between the techniques of [Sei] and [Wel]. (It is interesting, that there are examples of linear programs (with few constraints), where adding the move-to-front heuristic to our new algorithm gives a significantly worse performance [Mat1].) Our algorithm also seems to behave well in practice, and its analysis, as given here, also provides a considerably improved worst-case upper bound on its expected complexity. Recently, derandomization techniques have been applied to Clarkson's algorithm to obtain deterministic $O(C_d n)$ time LP algorithms with C_d of the order $2^{O(d \log d)}$ [ChM].

The abstract framework we present considers the set H of n constraints, and a function w which maps every subset of H to its optimal solution, where w satisfies a few simple conditions; as it turns out, this is all which is needed to prove the correctness of the algorithm, and to analyze its expected running time in terms of two primitive operations — 'violation tests' and 'basis computations'. It turns out that these primitive operations can easily be implemented in polynomial time for linear programming, but that is by no means clear for other instances of problems in the framework. For example in the case of computing the smallest enclosing ball, a basis computation amounts to computing the smallest ball enclosing $d+2$ points in d-space. Only recently, Gärtner [Gär2] was able to show that this can be done with expected $e^{O(\sqrt{d})}$ arithmetic operations.

Clarkson's algorithm [Cla3] can also be shown to work in this framework, while the algorithm of Seidel [Sei] (and the generalization to other applications in [Wel]) needs to make a more explicit use of the geometry of the problems. A different framework has been recently developed by Dyer [Dye2], which yields deterministic algorithms linear in n (with larger constants in d).

The paper is organized as follows. In the next section we introduce some notations and review basic observations on linear programming that are required for the presentation and analysis of our algorithm in Section 3. The analysis culminates in a recurrence relationship, whose solution is a nontrivial and interesting task in itself; Section 4 is devoted to that solution. Finally, in Section 5 we describe the abstract framework, and mention a few examples.

2 Notation and Basics

In the following we will prepare the notation required for our presentation. We model the case of linear programming where the objective function is defined by the vertical vector $c = (-1, 0, 0 \ldots, 0)$, and the problem is to maximize $c \cdot x$ over an intersection of given n halfspaces.

We let '$<$' be the lexicographical ordering on R^d, (i.e. $x < x'$, if $x_1 < x'_1$, or $x_1 = x'_1$

and $x_2 < x'_2$, or so on). This ordering is extended to $\mathrm{R}^d \cup \{+\infty, -\infty\}$ ($+\infty$ and $-\infty$ being special symbols) by the convention that $-\infty < x < +\infty$ for all $x \in \mathrm{R}^d$.

Let H be a set of closed halfspaces in d-space (which we call also *constraints in d-space*). By v_H we denote the lexicographically smallest point in the *feasible region* $F_H := \bigcap_{h \in H} h$; v_H is called the *value of* H. (It seems more standard to look for the *backwards* lexicographically smallest point, i.e., the case when $c = (0, 0, \ldots, 0, -1)$; the reader accustomed to this may prefer to think 'backwards' instead.) If F_H is empty, then we let $v_H = +\infty$. If F_H is nonempty, but contains no minimum, then we let $v_H = -\infty$.

Intuitively, we can view $+\infty$ as a point which lies in all halfspaces and which dominates all points. $-\infty$ may be seen as a point in minus infinity, or simply as a symbol for 'undefined'.

A *basis* B is a set of constraints with $-\infty < v_B$, and $v_{B'} < v_B$ for all proper subsets B' of B. If $-\infty < v_H$, a *basis of* H is a minimal subset B of H with $v_B = v_H$.

Constraint h is *violated by* H, if $v_H < v_{H \cup \{h\}}$; if $-\infty < v_H$, then this is equivalent to $v_H \notin h$. Constraint h is *extreme in* H, if $v_{H - \{h\}} < v_H$. Note that h is extreme in H if and only if $h \in H$ and h is violated by $H - \{h\}$.

The following properties are either trivial (like (i) and (ii)), or standard in linear programming theory (like (iii)). We cite them explicitly here, because they constitute all that is needed for the correctness and time analysis of the algorithms to be described.

Lemma 1 *Let G and H be sets of constraints in d-space, $G \subseteq H$, and let h be a constraint in d-space.*
(i) $v_G \leq v_H$.
(ii) *If $-\infty < v_G = v_H$, then h is violated by G if and only if h is violated by H.*
(iii) *If $v_H < +\infty$, then any basis $B \subseteq H$ has exactly d constraints, and any $F \subseteq H$ has at most d extreme constraints.* □

3 The Algorithm

Let H_+ be the set of constraints $x_i \geq 0$, for $1 \leq i \leq d$. Given a set H of constraints, the following algorithm will compute a basis of $H \cup H_+$; such a basis always exists, because $-\infty < (0, 0, \ldots, 0) \leq v_{H \cup H_+}$. (The algorithm works for any initial basis B_0 instead of H_+; in particular, we could take any basis $B_0 \subseteq H$, if available, or we can take a set of constraints $x_i \geq -\omega$, $1 \leq i \leq d$, for a symbolic value $-\omega$ representing an arbitrarily small number.) We will always assume that $v_{H \cup H_+} < +\infty$ for the constraint sets we consider. It is well-known that this condition can be obtained at the cost of an extra dimension and an extra constraint. However, note that we have not made any general position assumptions; so, e.g., we do not require that bases are unique.

Given a set H of n constraints, we might first consider the following trivial approach. Remove a random constraint h, and compute a basis B for $H - \{h\}$ recursively. If h is not violated by B (or equivalently, if h is not violated by $H - \{h\}$), then B is a basis for H and we are done. If h is violated, then we try again by removing a (hopefully different) random constraint. Note that the probability that h is violated by $H - \{h\}$ is at most $\frac{d}{n}$ since there are at most d extreme constraints in H.

So much for the basic idea, which we can already find in Seidel's LP algorithm [Sei]; however, in order to guarantee efficiency, we need some additional ingredients. First, the procedure `SUBEX_lp` actually solving the problem has two parameters: the set H of constraints, and a basis $C \subseteq H$; in general, C is not a basis of H, and we call it a *candidate basis*. It can be viewed as some auxiliary information we get for the computation of the solution. Note that C has no influence on the output of the procedure (but it influences its efficiency).

The problem of computing a basis of $H \cup H_+$ can now be solved by:

```
function procedure subex_lp(H);                 /* H: set of n constraints in d-space;
    B := SUBEX_lp(H ∪ H+, H+);                   /* returns a basis of H ∪ H+.
    return B;
```

The following pseudocode for `SUBEX_lp` uses two primitive operations: The first is a test 'h `is violated by` B', for a constraint h and a basis B (*violation test*); this operation can be implemented in time $O(d)$, if we keep v_B with the basis B. Second, `SUBEX_lp` assumes the availability of an operation, '`basis`(B,h)', which computes a basis of $B \cup \{h\}$ for a d-element basis B and a constraint h (*basis computation*). This step corresponds to a pivot step, and with an appropriate representation of B, it can be performed with $O(d^2)$ arithmetic operations. Note that if $-\infty < v_B$, then $-\infty < v_{B\cup\{h\}}$, and so $-\infty < v_H$ for all constraint sets H considered in an execution.

```
function procedure SUBEX_lp(H, C);              /* H: set of n constraints in d-space;
    if H = C then                                /* C ⊆ H: a basis;
        return C                                 /* returns a basis of H.
    else
        choose a random h ∈ H − C;
        B := SUBEX_lp(H − {h}, C);
        if h is violated by B then               /* ⇔ v_B ∉ h
            return SUBEX_lp(H, basis(B, h))
        else
            return B;
```

A simple inductive argument shows that the procedure returns the required answer. This happens after a finite number of steps, since the first recursive call decreases the number of constraints, while the second call increases the value of the candidate basis (and there are only finitely many different bases).

For the analysis of the expected behavior of the algorithm, let us take a closer look at the probability that the algorithm makes the second recursive call with candidate basis '`basis`(B,h)'. As noted above, this happens exactly when h is extreme in h. Since we now choose h from $H - C$ (and C always has d elements), it follows that the probability for h being extreme is at most $\frac{d}{n-d}$. Moreover, if $d-k$ extreme constraints in H are already in C, then the bound improves to $\frac{k}{n-d}$; in fact, there are never more 'bad' choices than there are choices at all, and so the bound can be lowered to $\frac{\min\{n-d,k\}}{n-d}$. We want to show that the numerator decreases rapidly as we go down the recursion, and this will entail the subexponential time bound.

The key notion in our analysis will be that of *hidden dimension*. For given $G \subseteq H$ and a basis $C \subseteq G$, the hidden dimension of the pair (G,C) measures how "close" is C to the solution (basis) of G. The hidden dimension of (G,C) is d minus the number of constraints $h \in C$ which must be contained in any basis $B \subseteq G$ with value greater than or equal to the value of C.

Lemma 2 *The hidden dimension of* (G, C) *is* $d - |\{h \in G;\ v_{G-\{h\}} < v_C\}|$.

Proof. We need to show that a constraint $h \in G$ satisfies $v_{G-\{h\}} < v_C$ iff h appears in all bases $B \subseteq G$ with $v_B \geq v_C$. If $v_{G-\{h\}} < v_C$ and $B \subseteq G$ is a basis with $v_B \geq v_C$, and h is not in B, then $B \subseteq G - \{h\}$, so $v_B \leq v_{G-\{h\}} < v_C$, a contradiction. Conversely, if $v_{G-\{h\}} \geq v_C$ then let B be a basis for $G - \{h\}$. We have $v_B = v_{G-\{h\}} \geq v_C$ and h is not in B. This completes the proof of the lemma. ◻

Let us remark that the hidden dimension of (G, C) depends on C only via the *value* of C. An intuitive interpretation is that *all* the "local optima" defined by constraints of G lying above the "level" v_C are contained in a k-flat defined by some $d - k$ bounding hyperplanes of constraints of C, k being the hidden dimension. Hidden dimension zero implies that C is a basis of G.

We enumerate the constraints in H in such a way that

$$v_{H-\{h_1\}} \leq v_{H-\{h_2\}} \leq \cdots \leq v_{H-\{h_{d-k}\}} < v_C \leq v_{H-\{h_{d-k+1}\}} \leq \cdots \leq v_{H-\{h_n\}} \ .$$

The ordering is not unique, but the parameter k emerging from this ordering is unique and, by definition, it equals the hidden dimension of (H, C).

Note that for $h \in H - C$, $v_C \leq v_{H-\{h\}}$, and so $h_1, h_2, \ldots, h_{d-k}$ are in C. Hence the only choices for h which possibly entail a second call are $h_{d-k+1}, \ldots, h_d$ (for $i > d$, $v_{H-\{h_i\}} = v_H$). Suppose that, indeed, a constraint h_{d-k+i}, $1 \leq i \leq k$, is chosen, and the first call (with candidate basis C) returns with basis B, $v_B = v_{H-\{h_{d-k+i}\}}$. Then we compute $C' = \texttt{basis}(B, h_{d-k+i})$. Since $v_{H-\{h_{d-k+i}\}} = v_B < v_{C'}$, the hidden dimension of the pair (H, C') for the second call is at most $k - i$.

The hidden dimension is monotone, i.e., if $C \subseteq G \subseteq H$, then the hidden dimension of (G, C) does not exceed the hidden dimension of (H, C). This holds, because the constraints $h_1, h_2, \ldots, h_{d-k}$ are in C (and so in G), and $v_{G-\{h_i\}} \leq v_{H-\{h_i\}}$ because $G \subseteq H$.

We denote by $t_k(n)$ and $b_k(n)$ the maximum expected number of violation tests and basis computations, respectively, entailed by a call `SUBEX_lp`(H, C) with n constraints and hidden dimension $\leq k$. The discussion above yields the following recurrence relationships: $t_k(d) = 0$ for $0 \leq k \leq d$,

$$t_k(n) \leq t_k(n-1) + 1 + \frac{1}{n-d} \sum_{i=1}^{\min\{n-d,k\}} t_{k-i}(n) \quad \text{for } n > d \ ,$$

and $b_k(d) = 0$ for $0 \leq k \leq d$,

$$b_k(n) \leq b_k(n-1) + \frac{1}{n-d} \sum_{i=1}^{\min\{n-d,k\}} (1 + b_{k-i}(n)) \quad \text{for } n > d. \tag{3.1}$$

Simple proofs by induction show that $b_k(n) \leq 2^k(n-d)$ and $t_k(n) \leq 2^k(n-d)$. It turns out that for n not much larger than d, this is a gross overestimate, and in the next section we will show that

$$b_d(n+d) \leq e^{2\sqrt{d \ln(n/\sqrt{d})} + O(\sqrt{d} + \ln n)} \ .$$

We have also $t_k(n) \leq (n-d)(1 + b_k(n))$; indeed, every constraint is tested at most once for violation by B, for each basis B ever appearing in the computation; we have tha factor $(n-d)$ since we do not test the elements in B for violation by B, and we add 1 to $b_k(n)$ to account for the initial basis H_+.

Recall that we account $O(d)$ arithmetic operations for a violation test, and $O(d^2)$ arithmetic operations for a basis computation. Note also that for the computation of $v_{H\cup H_+}$, we have to add the d nonnegativity constraints before we initiate `SUBEX_lp`. Anticipating the solution of the recurrences, given in the next section, we thus conclude:

Theorem 3 *Let H be a set of n constraints in d-space. The lexicographically smallest nonnegative point $v_{H\cup H_+}$ in the intersection of the halfspaces in H can be computed by a randomized algorithm with an expected number of*

$$O(nd+d^2)\,b_d(n+d) = e^{2\sqrt{d\ln(n/\sqrt{d})}+O(\sqrt{d}+\ln n)}$$

arithmetic operations. □

Clarkson [Cla3] shows that a linear program with n constraints in d-space can be solved with expected

$$O(d^2n + d^4\sqrt{n}\log n + T_d(9d^2)d^2\log n) \tag{3.2}$$

arithmetic operations, where $T_d(9d^2)$ stands for the complexity of solving a linear program with $9d^2$ constraints in d-space. We replace $T_d(9d^2)$ by our bound, and observe that, after this replacement, the middle term in (3.2) is always dominated by the two other terms; moreover, we can even omit the '$\log n$' factor in the last term in (3.3) without changing the asymptotics of the whole expression. Thus, we obtain:

Corollary 4 *The lexicographically smallest nonnegative point $v_{H\cup H_+}$ in the intersection of the n halfspaces H in d-space can be computed by a randomized algorithm with an expected number of*

$$O(d^2n + e^{O(\sqrt{d\log d})}) \tag{3.3}$$

arithmetic operations. □

4 Solving the Recurrence

In this section we prove the promised bounds for $b_k(n)$. We first put $\hat{b}_k(n) = 1 + b_k(n+d)$, and we let $f(k,n)$ be the function which satisfies the recurrence for $\hat{b}_k(n)$ with equality (thus majorizing $\hat{b}_k(n)$).

Thus $f(k,n)$ is the solution to the recurrence

$$f(k,0) = f(0,n) = 1 \qquad \text{for } k,n \geq 0,$$

and

$$f(k,n) = f(k,n-1) + \frac{1}{n}\sum_{j=1}^{\min(n,k)} f(k-j,n) \qquad \text{for } k,n \geq 1. \tag{4.1}$$

We prove the following upper bound:

Lemma 5 *For all n,k, $f(k,n) \leq 1 + 2^k n$ holds. If $n \leq e^{k/4}\sqrt{k}$, then*

$$f(k,n) \leq \exp\left(2\sqrt{k\ln\frac{n}{\sqrt{k}}} + O\left(\sqrt{k}+\ln n\right)\right).$$

For a large range of values of n, k, this bound is essentially tight (at least the leading term). In order to avoid various technicalities in the lower bound proof, we restrict the range of n and we will not try to get as tight a bound as we could (concerning the order of magnitude of the lower order terms). We show

Lemma 6 *For k tending to infinity and n such that $k \leq n \leq 2^{o(k)}$, we have*

$$f(k,n) \geq \exp\left((2-o(1))\sqrt{k \ln \frac{n}{\sqrt{k}}}\right) .$$

We emphasize that this is a lower bound on $f(k,n)$ and not on $\hat{b}_k(n)$. We do not have such a lower bound on $\hat{b}_k(n)$, and it is conceivable that $\hat{b}_k(n)$ is much smaller. See [Mat1] for recent related lower bounds.

Upper bound. For the *proof of Lemma* 5 we apply the technique of generating functions (the easy inductive proof of the '$1+2^k n$'-bound has been mentioned in the previous section, and will be omitted here). Define, for each $n \geq 0$,

$$G_n(z) = \sum_{k=0}^{\infty} f(k,n) z^k .$$

If we multiply the recurrence (4.1) by z^k and sum over k, we obtain

$$G_n(z) = G_{n-1}(z) + \frac{1}{n} \sum_{k=0}^{\infty} \sum_{j=1}^{\min(n,k)} f(k-j,n) z^k =$$

$$G_{n-1}(z) + \frac{1}{n} \sum_{j=1}^{n} \sum_{k=j}^{\infty} f(k-j,n) z^k =$$

$$G_{n-1}(z) + \frac{1}{n} \sum_{j=1}^{n} z^j \sum_{k=j}^{\infty} f(k-j,n) z^{k-j} =$$

$$G_{n-1}(z) + \left(\frac{1}{n} \sum_{j=1}^{n} z^j\right) \cdot G_n(z) .$$

In other words, we have the recurrence

$$G_n(z) = \frac{G_{n-1}(z)}{1 - \frac{1}{n}\sum_{j=1}^{n} z^j} .$$

Since $G_0(z) = \frac{1}{1-z}$, it follows that

$$G_n(z) = \frac{1}{1-z} \cdot \prod_{j=1}^{n} \frac{1}{1 - \frac{1}{j}\sum_{i=1}^{j} z^i} .$$

Regarding z as a complex variable, we want to find the value of the coefficient $f(k,n)$ of z^k in the Taylor expansion of $G_n(z)$. As is well known, this is equal to

$$f(k,n) = \frac{1}{2\pi i} \int_{\gamma} \frac{G_n(z)}{z^{k+1}} dz$$

where the integral is over a closed curve γ that contains the origin and does not contain any other pole of $G_n(z)$.

We will choose for γ the circle $|z| = t$, for $t < 1$. It is easily checked that none of the denominators in the product defining $G_n(z)$ can vanish for $|z| < 1$: this follows from the inequality

$$|\frac{1}{j}\sum_{i=1}^{j} z^i| \le \frac{1}{j}\sum_{i=1}^{j} |z|^i < 1 .$$

This also implies that

$$|1 - \frac{1}{j}\sum_{i=1}^{j} z^i| \ge 1 - |\frac{1}{j}\sum_{i=1}^{j} z^i| \ge 1 - \frac{1}{j}\sum_{i=1}^{j} |z|^i .$$

which yields a bound of

$$f(k,n) \le \frac{1}{t^k(1-t)} \cdot \prod_{j=1}^{n} \frac{1}{F_j} , \tag{4.2}$$

where

$$F_j = 1 - \frac{t + t^2 + \cdots t^j}{j} = 1 - \frac{t - t^{j+1}}{j(1-t)} .$$

Let $q \ge 2$ be an integer parameter to be determined later, and let us set

$$t = 1 - \frac{1}{q} .$$

We estimate the terms in the product for $q > 2$. We distinguish two cases. First, for $j \ge q$ we use the estimate

$$F_j \ge 1 - \frac{t}{j(1-t)} = 1 - \frac{q-1}{j} = \frac{j-q+1}{j} ,$$

so, using Stirling's formula,

$$\prod_{j=q}^{n} \frac{1}{F_j} \le \frac{q \cdot (q+1) \cdots n}{(n-q+1)!} = \frac{(n-q+2) \cdots n}{(q-1)!} \le \left(\frac{3n}{q}\right)^q .$$

For $j < q$, we calculate

$$F_j = \frac{\frac{j}{q} - 1 + \frac{1}{q} + (1 - \frac{1}{q})^{j+1}}{j/q} = \frac{1}{j}\left(\binom{j+1}{2} q^{-1} - \binom{j+1}{3} q^{-2} + \ldots\right) .$$

Since $j < q$, the absolute values of the terms in the alternating sum in the parentheses form a decreasing sequence. Hence if we stop the summation after some term, the error will have the same sign as the first omitted term. So we have

$$F_j \ge \frac{1}{j}\left(\frac{(j+1)j}{2q} - \frac{(j+1)j(j-1)}{6q^2}\right) \ge \frac{j+1}{2q} - \frac{j^2}{6q^2} \ge \frac{j}{3q} .$$

Then

$$\prod_{j=1}^{q-1} \frac{1}{F_j} \le \frac{(3q)^q}{q!} \le 9^q . \tag{4.3}$$

Finally we estimate

$$t^k = \left(1 - \frac{1}{q}\right)^k \geq (e^{-1/q-1/q^2})^k = e^{-k/q-k/q^2}\,,$$

so we get

$$f(k,n) \leq q e^{k/q+k/q^2} 27^q (n/q)^q = \exp(\frac{k}{q} + \frac{k}{q^2} + q\ln(n/q) + O(q))\,.$$

For most values of n, k, we set

$$q = \left\lfloor \sqrt{\frac{k}{\ln \frac{n}{\sqrt{k}}}} \right\rfloor\,,$$

which yields the bound

$$f(k,n) \leq \exp\left(2\sqrt{k \ln \frac{n}{\sqrt{k}}} + O\left(\sqrt{k} + \ln n\right)\right)\,. \tag{4.4}$$

There are two extreme cases. If the above definition of q gives $q < 2$ (this happens when n is exponentially large compared to k, $n > e^{k/4}\sqrt{k}$), we use the easy bound of $f(k,n) \leq 1 + 2^k n = e^{O(\ln n)}$. And if n comes close to $\sqrt{k}$ or it is smaller, we set $q = \sqrt{k}$ and obtain the bound $e^{O(\sqrt{k})}$. (By playing with the estimates somewhat further, one can get better bounds from (4.2) both for $n << \sqrt{k}$ (in this case we observe that that if $n < q$, the product in (4.3) only goes up to n, so that one gets C^n instead of C^q, and then a better choice of q is possible), and for n very large, that is, $n >> e^{k/4}\sqrt{k}$.) This establishes Lemma 5. ◻

Lower bound. The proof of the lower bound is based on the following "explicit form" for $f(k,n)$:

Lemma 7

$$f(k,n) = \sum_{(m_1,\ldots,m_q)} \sum_{(i_1,\ldots,i_q)} \frac{1}{i_1 i_2 \ldots i_q}\,, \tag{4.5}$$

where the first sum is over all q-tuples $(m_1,\ldots,m_q)$, $0 \leq q \leq k$, *with* $m_j \geq 1$ *and* $m_1 + m_2 + \cdots + m_q \leq k$, *and the second sum is over all q-tuples* $(i_1,\ldots,i_q)$ *with* $1 \leq i_1 \leq i_2 \leq \ldots \leq i_q \leq n$ *and with* $i_j \geq m_j$ *for every* $j = 1, 2, \ldots, q$. *(Here q can also be 0; this contributes one term equal to 1 to the sum. We can formally interpret this as a term corresponding to the (unique) 0-tuple; this term, as the empty product, is equal to 1.)*

Proof. As for the initial conditions, for $k = 0$ we only have the term with $q = 0$ in the sum, yielding the value 1. Similarly for $n = 0$, we only get a nonzero term for $q = 0$, which again gives 1.

Consider the difference $f(k,n) - f(k,n-1)$, where $f(k,n)$ is defined by (4.5). The terms which appear in (4.5) for k, n and not for $k, n-1$ are precisely those with $i_q = n$. For the sum of such terms, we have

$$\sum_{\substack{m_1+\cdots+m_q \leq k \\ m_j \geq 1}} \sum_{\substack{i_1 \leq \ldots i_{q-1} \leq i_q = n \\ i_j \geq m_j}} \frac{1}{i_1 i_2 \ldots i_q} =$$

$$\sum_{m_q=1}^{\min(n,k)} \frac{1}{n} \sum_{\substack{m_1+\cdots+m_{q-1}\le k-m_q \\ m_j\ge 1}} \sum_{\substack{i_1\le\ldots\le i_{q-1}\le n \\ i_j\ge m_j}} \frac{1}{i_1 i_2 \ldots i_{q-1}} =$$

$$\frac{1}{n} \sum_{j=1}^{\min(n,k)} f(k-j,n)\,.$$

The function defined by (4.5) thus satisfies both the initial conditions and the recurrence relation. ◻

For the *proof of Lemma* 6 let us first define several auxiliary parameters. Let $\varepsilon = \varepsilon(k)$ be a function tending to 0 slowly enough, for instance $\varepsilon(k) = 1/\log\log k$. Set

$$q = \sqrt{k \ln \frac{n}{\sqrt{k}}}\,, \quad m = \frac{k}{\varepsilon q} + 1\,.$$

Since we assume that k is large, we may neglect the effect of truncating q and m to integers; similarly for other integer parameters in the sequel. The assumption $\log n = o(k)$ guarantees that $q = o(k)$.

Consider the inner sum in (4.5) for a given q-tuple $(m_1, \ldots, m_q)$ with $1 \le m_j \le m$ for all $j = 1, 2, \ldots, q$ and $m_1 + m_2 + \cdots + m_q \le k$. Then this sum is at least

$$\sum_{m\le i_1\le i_2\le\ldots\le i_q\le n} \frac{1}{i_1 i_2 \ldots i_q} \ge \frac{1}{q!} \sum_{m\le i_1,\ldots,i_q\le n} \frac{1}{i_1 i_2 \ldots i_q} =$$

$$\frac{1}{q!}\left(\frac{1}{m} + \frac{1}{m+1} + \cdots + \frac{1}{n}\right)^q\,.$$

One easily verifies (by induction, say) that $\frac{1}{m} + \frac{1}{m+1} + \cdots + \frac{1}{n} \ge \ln(n/m)$. Furthermore, we use Stirling's formula to get the estimate $q! = (q/e)^{q(1+o(1))}$, so we can estimate the above expression from below by

$$\left(\frac{e\ln(n/m)}{q}\right)^{q(1-o(1))}\,. \tag{4.6}$$

It is well-known (and easy to prove) that the number $N(r,k)$ of r-tuples $(m_1, \ldots, m_r)$ with $m_j \ge 1$ and $m_1 + m_2 + \cdots + m_r \le k$ is $\binom{k}{r}$. Let $N^*(r,k,m)$ denote the number of r-tuples as above with the additional condition $m_j \le m$; we claim that if $q \ge r + k/(m-1)$ then

$$N^*(q,k,m) \ge N(r,k-q)\,. \tag{4.7}$$

To see this it suffices to encode every r-tuple $(m_1, \ldots, m_r)$ contributing to $N(r,k-q)$ by a q-tuple contributing to $N^*(q,k,m)$. Such an encoding can be performed as follows: if $m_j \le m-1$ we leave this element as it is, otherwise we replace it by $\lfloor m_j/(m-1)\rfloor$ elements equal to m and one element equal to $m_j+1-(m-1)\lfloor m_j/(m-1)\rfloor \in [1, m-1]$ (for instance, with $m = 3$, the tuple $(1,5,2,8)$ will be transformed to $(1,3,3,2,2,3,3,3,3,1)$; note that if m_j is replaced by a block of t elements, their sum is $m_j + t$). The number of elements of the resulting vector is $\sum_{j=1}^{r}(1 + \lfloor m_j/(m-1)\rfloor) \le r + (k-q)/(m-1) \le q$, and we may then pad the vector with ones to get exactly q elements. It is easy to see that this encoding is reversible. By what has just been observed, the sum of elements in the resulting vector will exceed the sum of the initial r-tuple by at most q. This proves (4.7).

With our value of m, we may choose $r = q(1-\varepsilon)$ and use (4.7) to show that the number of q-tuples $(m_1, \dots, m_q)$ with $1 \le m_j \le m$ and $m_1 + \cdots + m_q \le k$ is at least

$$N(q(1-\varepsilon), k-q) = \binom{k-q}{q(1-\varepsilon)} \ge \frac{(k-2q)^{q(1-\varepsilon)}}{(q(1-\varepsilon))!} \ge \left(\frac{ke}{q}\right)^{q(1-o(1))} .$$

Combining this with (4.6) and observing that $\ln(n/m) \approx \ln(n\varepsilon q/k) \ge \ln(n/\sqrt{k})$, we obtain

$$f(k,n) \ge \left(\frac{e^2 k \ln \frac{n}{\sqrt{k}}}{q^2}\right)^{q(1-o(1))} = e^{2q(1-o(1))} ,$$

which proves Lemma 6. □

5 An Abstract Framework

Let us consider optimization problems specified by pairs (H, w), where H is a finite set, and $w : 2^H \to \mathcal{W}$ is a function with values in a linearly ordered set $(\mathcal{W}, \le)$; we assume that $\mathcal{W}$ has a minimum value $-\infty$. The elements of H are called *constraints*, and for $G \subseteq H$, $w(G)$ is called the *value of* G. The goal is to compute a minimal subset B_H of H with the same value as H (from which, in general, the value of H is easy to determine), assuming the availability of two basic operations to be specified below. It turns out that the algorithm in Section 3 can be used to perform this computational task, as long as the following axioms are satisfied:

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

$$w(F) \le w(G).$$

Axiom 2. (*Locality*) For any $F \subseteq G \subseteq H$ with $-\infty < w(F) = w(G)$ and any $h \in H$,

$$w(G) < w(G \cup \{h\}) \text{ implies that also } w(F) < w(F \cup \{h\}).$$

If Axioms 1 and 2 hold, then we call (H, w) an *LP-type problem*. Obviously, linear programming is an LP-type problem, if we set $w(G) = v_G$ for a constraint set $G \subseteq H$. Then the axioms coincide with Lemma 1 (i) and (ii). The notions needed in Section 3 carry over in the obvious way: A *basis* B is a set of constraints with $-\infty < w(B)$, and $w(B') < w(B)$ for all proper subsets B' of B. For $G \subseteq H$, if $-\infty < w(G)$, a *basis of* G is a minimal subset B of G with $w(B) = w(G)$. Constraint h is *violated by* G, if $w(G) < w(G \cup \{h\})$. Constraint h is *extreme in* G, if $w(G - \{h\}) < w(G)$.

For the efficiency of the algorithm the following parameter is crucial: The maximum cardinality of any basis is called the *combinatorial dimension of* (H, w), and is denoted by $\dim(H, w)$.

We assume that the following primitive operations are available.

(*Violation test*) 'h `is violated by` B', for a constraint h and a basis B, tests whether h is violated by B or not.

(*Basis computation*) '`basis`(B, h)', for a constraint h and a basis B, computes a basis of $B \cup \{h\}$.

(Note that checking whether h violates B is equivalent to checking whether $w(\texttt{basis}(B,h)) > w(B)$. This shows that the two primitive operations are closely related.) Now we have all the ingredients necessary to apply our algorithm to an LP-type problem, provided we have an initial basis B_0 for the first call of `SUBEX_lp`. We can also show, using the simpler inductive argument mentioned in Section 3, that the expected number of primitive operations performed by the algorithm is $O(2^\delta n)$, where $n = |H|$ and $\delta = \dim(H,w)$ is the combinatorial dimension. However, in order to ensure the subexponential time bound we need an extra condition:

> **Axiom 3.** (*Basis regularity*) If B is a basis with $|B| = \dim(H,w)$, and h is a constraint, then every basis of $B \cup \{h\}$ has exactly $\dim(H,w)$ elements.

If Axioms 1-3 are satisfied, then we call (H,w) a *basis-regular LP-type problem.* We have seen that linear programming in d-space is a basis-regular LP-type problem of combinatorial dimension d, provided the program is feasible (Lemma 1 (iii) provides the stronger property that every basis has the same cardinality). Actually, in the whole treatment in Section 3 we were careful not to use other properties of linear programming except for those formulated in Lemma 1. (We would be very glad to use any other properties to obtain a faster algorithm, but we do not know how.) Of course, we have an extra computational assumption in order to start the algorithm.

> (*Initial basis*) An initial basis B_0 with exactly $\dim(H,w)$ elements is available.

(In the case of linear programming, the nonnegativity-constraints H_+ can play the role of the initial basis.) We may conclude

Theorem 8 *Given a basis-regular LP-type problem (H,w), $|H| = n$, of combinatorial dimension δ and an initial basis $B_0 \subseteq H$, $|B_0| = \dim(H,w)$, the algorithm* `SUBEX_lp` *computes a basis of H with an expected number of at most*

$$e^{2\sqrt{\delta \ln((n-\delta)/\sqrt{\delta})}+O(\sqrt{\delta}+\ln n)}$$

violation tests and basis computations. $\Box$

As it turns out, Clarkson's algorithm can also be formulated and analysed within the framework, and, if the basic cases (each involving $O(\delta^2)$ constraints) are solved by our algorithm, then the expected number of required violation tests is $O(\delta n + e^{O(\sqrt{\delta \log \delta})})$, and the expected number of required basis computations is $e^{O(\sqrt{\delta \log \delta})} \log n$.

Many problems can be shown to satisfy Axioms 1 and 2 (see below for a list of such problems), but — except for linear programming — basis-regularity is not naturally satisfied by any of them. However, we can artificially enforce Axiom 3 by the following trick (due to Bernd Gärtner, [Gär1]): Let (H,w) be an LP-type problem of combinatorial dimension δ and with value set $\mathcal{W}$. Then we define a pair (H,w'), where

$$w'(G) \quad = \quad \begin{cases} -\infty & \text{if } w(G) = -\infty, \text{ and} \\ (w(G), \min\{\delta, |G|\}) & \text{otherwise} \end{cases} \tag{5.1}$$

for $G \subseteq H$, and the new value set is $\{-\infty\} \cup (\mathcal{W} - \{-\infty\}) \times \{0,1,2,\ldots,\delta\}$ with lexicographical ordering. The straightforward proof of the following lemma is omitted.

Lemma 9 *Given an LP-type problem (H, w), the pair (H, w') as defined above is a basis-regular LP-type problem of combinatorial dimension* $\dim(H, w)$. *Any basis B complemented by arbitrary* $\dim(H, w) - |B|$ *elements can serve as initial basis.* ◻

Hence, we can transform every LP-type problem into a basis-regular one, although we have to be careful about the new interpretation of violation tests and basis computations. We thus obtain an algorithm with an expected subexponential number of violation tests and basis computations, but those primitive operations might be quite expensive. We exhibit now two examples of LP-type problems where we can successfully apply our algorithm (including efficient implementation of the primitive operations).

Smallest enclosing ball. The problem of computing the disk of smallest radius containing a given set of n points in the plane goes back to J. J. Sylvester in 1857 [Syl]. The first linear time solution to this problem was provided by Megiddo [Meg1] (see this reference for a short guide through previous $O(n^3)$ and $O(n \log n)$ solutions). The general problem of computing the smallest ball enclosing a set of n points in d-space can be solved in linear time as long as the dimension is considered fixed, see [Meg1, Meg2] for a deterministic algorithm, and [Wel] for a simple randomized algorithm; however both algorithms are exponential in d.

Given a set P of n points in d-space, we define $r(Q)$ as the smallest radius of a closed ball containing $Q \subseteq P$. It is well-known that this smallest radius exists, and that the ball realizing this radius is unique. Moreover, there is always a subset B of Q containing at most $d + 1$ points with $r(B) = r(Q)$ (see, e.g. [Jun]). With these basic facts in hand, it is easy to show that (P, r) is an LP-type problem of combinatorial dimension $d + 1$: Clearly, adding points (*constraints*) to a set cannot decrease the radius of the smallest enclosing ball (and so monotonicity holds), and p is violated by $Q \subseteq P$ if and only of p lies outside the *unique* smallest ball containing Q (this easily implies locality). The problem is not basis-regular, so we have to apply the above transformation to validate our analysis. While the violation test is an easy 'point in ball'-test, the basis computation amounts to a non-trivial task: basically we have to compute the smallest ball enclosing $d+2$ points in d-space. Recently, Gärtner [Gär2] was able to show that this problem can be solved with expected $e^{O(\sqrt{d})}$ arithmetic operations. Hence, we obtain

Corollary 10 *The smallest enclosing ball of n points in d-space can be computed with an expected number of*

$$\min\{2^{d+O(\sqrt{d})}n, e^{2\sqrt{d \ln(n/\sqrt{d})}+O(\sqrt{d}+\ln n)}\}$$

arithmetic operations. ◻

Combining this with Clarkson's algorithm, the complexity reduces to the bound in (3.3) for linear programming.

Distance between polytopes. Given two closed polytopes $\mathcal{P}_1$ and $\mathcal{P}_2$, we want to compute their distance, i.e.

$$\operatorname{dist}(\mathcal{P}_1, \mathcal{P}_2) := \min\{\operatorname{dist}(a, b); a \in \mathcal{P}_1, b \in \mathcal{P}_2\},$$

with $\operatorname{dist}(a, b)$ denoting the Euclidean distance for points a and b. If the polytopes intersect, then this distance is 0. If they do not intersect, then this distance equals the maximum distance between two parallel hyperplanes separating the polytopes; such a pair (g, h) of hyperplanes is unique, and they are orthogonal to the segment connecting two points $a \in \mathcal{P}_1$ and $b \in \mathcal{P}_2$ with $\mathrm{dist}(a, b) = \mathrm{dist}(\mathcal{P}_1, \mathcal{P}_2)$. It is now an easy exercise to prove that there are always sets B_1 and B_2 of vertices of $\mathcal{P}_1$ and $\mathcal{P}_2$, respectively, such that $\mathrm{dist}(\mathcal{P}_1, \mathcal{P}_2) = \mathrm{dist}(\mathrm{conv}(B_1), \mathrm{conv}(B_2))$ (where $\mathrm{conv}(P)$ denotes the convex hull of a point set P), and $|B_1| + |B_2| \leq d + 2$; if the distance is positive, a bound of $d + 1$ holds.

We formulate the problem as an LP-type problem as follows: Let $V = V_1 \cup V_2$, where V_i is the vertex set of $\mathcal{P}_i$, $i = 1, 2$; we assume that $V_1 \cap V_2 = \emptyset$, so every subset U of V has a unique representation as $U = U_1 \cup U_2$, with $U_i \subseteq V_i$ for $i = 1, 2$. With this convention, we define for $U \subseteq V$, $w(U) = \mathrm{dist}(\mathrm{conv}(U_1), \mathrm{conv}(U_2))$; if U_1 or U_2 is empty, then we define $w(U) = \infty$. The pair (V, w) constitutes now an LP-type problem, except that the inequalities go the other way round: $U \subseteq W \subseteq V$ implies that $w(U) \geq w(W)$, and ∞ plays the role of $-\infty$. In order to see locality, observe that p is violated by U, if and only if p lies between the unique pair of parallel hyperplanes which separate U_1 and U_2 and have distance $w(U)$; this also shows how we can perform a violation test. From what we mentioned above, the combinatorial dimension of this problem is at most $d+2$ (or $d+1$, if the polytopes do not intersect). Hence, for a basis computation, we have to compute the distance between two polytopes in d space with altogether $d+3$ vertices. Again, we invoke [Gär2] to ensure that this can be performed with expected $e^{O(\sqrt{d})}$ arithmetic operations.

Corollary 11 *The distance between two convex polytopes in d-space with altogether n vertices can be computed with an expected number of*

$$\min\{2^{d+O(\sqrt{d})}n, e^{2\sqrt{d\ln(n/\sqrt{d})}+O(\sqrt{d}+\ln n)}\}$$

arithmetic operations. ⊡

The best previously published result was an expected $O(n^{\lfloor d/2 \rfloor})$ randomized algorithm (d considered fixed) in [Cla2]. The result in Corollary 11 can also be established if the polytopes are given as intersections of n halfspaces, and again combining the result with Clarkson's yields the bound in (3.3).

Other examples. There is quite a number of other examples which fit into the framework, and thus can be solved in time linear in the number of constraints (the dimension considered fixed). As we have mentioned before, the subexponential bound is a delicate issue, which depends how efficiently we can solve small problems. We just provide a list of examples, without giving further details.

> (*Smallest enclosing ellipsoid*) Given n points in d-space, compute the ellipsoid of smallest volume containing the points (combinatorial dimension $d(d+3)/2$, see [DLL, Juh, Wel]).

The problem has been treated in a number of recent papers, [Pos, Wel, Dye2, ChM]. Here the constraints are the points, and the value of a set of points is the volume of the smallest enclosing ellipsoid (in its affine hull). Such an ellipsoid (also called Löwner-John ellipsoid) is known to be unique, [DLL], from which locality easily follows; monotonicity is obviously satisfied. The primitive operations can be treated by applying general methods for solving systems of polynomial inequalities, but we cannot claim any subexponential time bounds (of course, the bound linear in the number of points holds).

(*Largest ellipsoid in polytope*) Given a polytope in d-space as the intersection of n halfspaces, compute the ellipsoid of largest volume contained in the polytope (combinatorial dimension $d(d+3)/2$).

(*Smallest intersecting ball*) Given n closed convex objects in d-space, compute the ball of smallest radius that intersects all of them (combinatorial dimension $d+1$).

In order to see the combinatorial dimension, we make the following observation. We consider the Minkowski sum of each convex object with a closed ball of radius r (centered at the origin). Then there is a ball of radius r intersecting all objects, if and only if the intersection of all of these Minkowski sums is nonempty. By Helly's Theorem, this intersection is nonempty, if and only if the intersection of any $d+1$ of them is nonempty. If r is the smallest radius which makes this intersection nonempty, then the interiors of the Minkowski sums do not have a common point, and so there is a set B' of at most $d+1$ of them which do not have a common point. The corresponding set of objects contains a basis (which is of cardinality at most $d+1$), and the claimed combinatorial dimension now follows easily.

(*Angle-optimal placement of point in polygon*) Let P be a star-shaped polygon with n vertices in the plane (a polygon is *star-shaped* if there is a point inside the polygon which sees all edges and vertices; the locus of all such points is called the *kernel*). We want to compute a point p in the kernel of P, such that after connecting p to all the vertices of P by straight edges, the minimal angle between two adjacent edges is maximized (combinatorial dimension 3).

Unlike in the previous examples, it might not be obvious what the constraints are in this problem. Let us assume that a polygon allows a placement of p with all angles occurring at least α. This restricts the locus of p to the intersection of the following convex regions: (i) for every vertex v of the polygon, and its incident edges e' and e with inner (with respect to P) angle β, there is a wedge of angle $\beta - 2\alpha$ with apex v, where p is forced to lie. (ii) For every edge e of P with incident vertices v and v', there is circular arc, which contains all points which see vertices v and v' at an angle α, and which lie on the inner side of e; point p is forced to lie in the region bounded by e and this circular arc. This suggests that we take the vertices (with incident edges) and the edges (with incident vertices) as constraints (for the purpose of the algorithm ignoring that they stem from a polygon). Thus the number of constraints is $2n$ and the combinatorial dimension is 3 by a reasoning using Helly's theorem as before.

(*Integer linear programming*) Given n halfspaces and a vector c in d-space, compute the point x with integer coordinates in the intersection of the halfspaces which maximizes $c \cdot x$ (combinatorial dimension 2^d, see [d'Oi, Bel, Sca]).

Although integer linear programming fits into the framework, it is a bad example in the sense that the basis computation has no bound in the unit cost model. There are some other problems which we did not mention so far, but may occur to the reader as natural examples, e.g.,

(*Largest ball in polytope*) Given a polytope in d-space as the intersection of n halfspaces, compute the ball of largest radius contained in the polytope.

(*Smallest volume annulus*) Given n points in d-space, find two concentric balls such that their symmetric difference contains the points and has minimal volume.

Indeed, these problems are LP-type problems, but actually they can be directly formulated as linear programs with $d+1$ and $d+2$, respectively, variables (the transformation for the smallest volume annulus-problem can be found in [Dör]). Thus also the subexponential time bounds hold.

Recently, Chazelle and Matoušek, [ChM], gave a deterministic algorithm solving LP-type problems in time $O(\delta^{O(\delta)}n)$, provided an additional axiom holds (together with an additional computational assumption). Still these extra requirements are satisfied in many natural LP-type problems. Matoušek, [Mat2], investigates the problem of finding the best solution satisfying all but k of the given constraints, for abstract LP-type problems as defined here.

Nina Amenta, [Ame1], considers the following extension of the abstract framework: Suppose we are given a family of LP-type problems (H, w_λ), parameterized by a real parameter λ; the underlying ordered value set $\mathcal{W}$ has a maximum element $+\infty$ representing *infeasibility*. The goal is to find the smallest λ for which (H, w_λ) is feasible, i.e. $w_\lambda(H) < +\infty$. [Ame1] provides conditions under which the such a problem can be transformed into a single LP-type problem, and she gives bounds on the resulting combinatorial dimension. Related work can be found in [Ame2].

6 Discussion

We have presented a randomized subexponential algorithm which solves linear programs and other related problems. Clearly, the challenging open problem is to improve on the bounds provided in [Kal] and here, and to find a polynomial combinatorial algorithm for linear programming.

In Section 4 we have shown that the bound we give is tight for the recursion (3.1) we derived from the analysis. Even stronger, [Mat1] gives abstract examples of LP-type problems of combinatorial dimension d and with $2d$ constraints, for which our algorithm requires $\Omega(e^{\sqrt{2d}}/\sqrt[4]{d})$ primitive operations. That is, in order to show a better bound of our algorithm for linear programming, we have to use properties other than Axioms 1 through 3.

Rote, [Rot], and Megiddo, [Meg3], suggest dual 'one-permutation' variants of the algorithm. It is interesting, that there are examples of linear programs, where a 'one-permutation' variant of the algorithm suggested in [Mat1] seems to behave much worse on certain linear programs than the original algorithm (this fact is substantiated only by experimental results, [Mat1]); this has to be seen in contrast to the situation for Seidel's linear programming algorithm, [Wel].

Acknowledgments. The authors would like to thank Gil Kalai for providing a draft copy of his paper, and Nina Amenta, Boris Aronov, Bernard Chazelle, Ken Clarkson, Bernd Gärtner, Nimrod Megiddo and Günter Rote for several comments and useful discussions.

References

[Ame1] N. Amenta, Helly-type theorems and generalized linear programming, *Discrete Comput. Geom.* **12** (1994) 241–261.

[Ame2] N. Amenta, Bounded boxes, Hausdorff distance, and a new proof of an interesting Helly-type theorem, *in* "Proc. 10th Annual ACM Symposium on Computational Geometry" (1994) 340–347.

[Bel] D. E. Bell, A theorem concerning the integer lattice, *Stud. Appl. Math.* **56** (1977) 187–188.

[ChM] B. Chazelle and J. Matoušek, On linear-time deterministic algorithms for optimization problems in fixed dimensions, Technical Report B 92–18, Dept. of Mathematics, Freie Universität Berlin (1992); also *in* "Proc. 4th SIAM-ACM Symposium on Discrete Algorithms" (1993) 281–290.

[Cla1] K. L. Clarkson, Linear programming in $O(n \times 3^{d^2})$ time, *Information Processing Letters* **22** (1986) 21–24.

[Cla2] K. L. Clarkson, New applications of random sampling in computational geometry, *Discrete Comput. Geom.* **2** (1987) 195–222.

[Cla3] K. L. Clarkson, Las Vegas algorithms for linear and integer programming when the dimension is small, manuscript (1989). Preliminary version: *in* "Proc. 29th IEEE Symposium on Foundations of Computer Science" (1988) 452–457, to appear in *J. Assoc. Comput. Mach.*

[DLL] L. Danzer, D. Laugwitz and H. Lenz, Über das Löwnersche Ellipsoid und sein Analogon unter den einem Eikörper eingeschriebenen Ellipsoiden, *Arch. Math.* **8** (1957) 214–219.

[Dan] D.B. Dantzig, *Linear Programming and Extensions,* Princeton University Press, Princeton, N.J. (1963).

[Dör] J. Dörflinger, Approximation durch Kreise: Algorithmen zur Ermittlung der Formabweichung mit Koordinatenmeßgeräten, Diplomarbeit, Universiät Ulm (1986).

[Dye1] M. E. Dyer, On a multidimensional search technique and its application to the Euclidean one-center problem, *SIAM J. Comput.* **15** (1986) 725–738.

[Dye2] M. E. Dyer, A class of convex programs with applications to computational geometry, *in* "Proc. 8th Annual ACM Symposium on Computational Geometry" (1992) 9–15.

[DyF] M. E. Dyer and A. M. Frieze, A randomized algorithm for fixed-dimensional linear programming, *Mathematical Programming* **44** (1989) 203–212.

[Gär1] B. Gärtner, personal communication (1991).

[Gär2] B. Gärtner, A subexponential algorithm for abstract optimization problems, *in* "Proc. 33rd IEEE Symposium on Foundations of Computer Science" (1992) 464–472, to appear in *SIAM J. Comput.*

[Juh] F. Juhnke, Volumenminimale Ellipsoidüberdeckungen, *Beitr. Algebra Geom.* **30** (1990) 143–153.

[Jun] H. Jung, Über die kleinste Kugel, die eine räumliche Figur einschließt, *J. Reine Angew. Math.* **123** (1901) 241–257.

[Kal] G. Kalai, A subexponential randomized simplex algorithm, *in* "Proc. 24th ACM Symposium on Theory of Computing" (1992) 475–482.

[Kar] N. Karmarkar, A new polynomial-time algorithm for linear programming, *Combinatorica* **4** (1984) 373–395.

[Kha] L. G. Khachiyan, Polynomial algorithm in linear programming, *U.S.S.R. Comput. Math. and Math. Phys.* **20** (1980) 53–72.

[KlM] V. Klee and G.J. Minty, How good is the simplex algorithm, *Inequalities III*, pp. 159–175, O. Shisha (ed.) Academic Press, New-York (1972).

[Mat1] J. Matoušek, Lower bounds for a subexponential optimization algorithm, *Random Structures & Algorithms* **5** (1994) 591-607.

[Mat2] J. Matoušek, On geometric optimization with few violated constraints, "Proc. 10th Annual ACM Symposium on Computational Geometry" (1994) 312–321.

[Meg1] N. Megiddo, Linear time algorithms for linear time programming in R^3 and related problems, *SIAM J. Comput.* **12** (1983) 759–776.

[Meg2] N. Megiddo, Linear programming in linear time when the dimension is fixed, *J. Assoc. Comput. Mach.* **31** (1984) 114–127.

[Meg3] N. Megiddo, A note on subexponential simplex algorithms, manuscript (1992).

[d'Oi] J.-P. d'Oignon, Convexity in cristallographic lattices, *J. Geom.* **3** (1973) 71–85.

[Pos] M. J. Post, Minimum spanning ellipsoids, *in* "Proc. 16th Annual ACM Symposium on Theory of Computing" (1984) 108–116.

[Rot] G. Rote, personal communication (1991).

[Sca] H. E. Scarf, An observation on the stucture of production sets with indivisibilities, *in* "Proc. of the National Academy of Sciences of the United States of America", **74** (1977) 3637–3641.

[Sei] R. Seidel, Low dimensional linear programming and convex hulls made easy, *Discrete Comput. Geom.* **6** (1991) 423–434.

[ShW] M. Sharir and E. Welzl, A combinatorial bound for linear programming and related problems, *in* "Proc. 9th Symposium on Theoretical Aspects of Computer Science", *Lecture Notes in Computer Science* **577** (1992) 569–579.

[Syl] J. J. Sylvester, A question in the geometry of situation, *Quart. J. Math.* **1** (1857) 79.

[Wel] E. Welzl, Smallest enclosing disks (balls and ellipsoids), *in* "New Results and New Trends in Computer Science", (H. Maurer, ed.), *Lecture Notes in Computer Science* **555** (1991) 359–370.