6 Lowdimensional Optimization

This chapter presents a general technique for solving linear optimization problems with d variables and n constraints with

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

arithmetic operations on expectation. This bound is based on two algorithms by Clarkson [1] and a subexponential algorithm by Kalai and by Matousek, Sharir, and Welzl [3]. On top of that, we will also consider LP-type problems and abstract optimization problems (such as the polytope distance and smallest enclosing circle) that can be solved with the techniques presented in this chapter. We will start with some basic definitions.

6.1 Definitions

We consider minimization problems with a set $S = \{x \in \mathbb{R}^d_+ \mid Ax \leq b\}$ of feasible solutions with $A \in \mathbb{R}^{(n,d)}$ and $b \in \mathbb{R}^n$ and objective function $f(x) = c^T x$ with $c > 0$, i.e. all coefficients in c are positive. The conditions $x \geq 0$ and $c > 0$ simplify our presentation since in this case there is always a bounded (or no) solution. But they can also be avoided by appropriate extensions of the algorithms. A vector $v = (v_1, \ldots, v_d)$ is the *lexicographically smallest* in a set M, if for all other vectors $w = (w_1, \ldots, w_d) \in M$ there exists an $i \in \{1, \ldots, d\}$ with $v_j = w_j$ for all $j < i$ and $v_i < w_i$.

Let H be the set of the n half spaces that are defined by the constraints $Ax \leq b$ and let H_+ be the set of the d half spaces that are defined by the constraint $x \geq 0$. For $G \subseteq H \cup H_+$ let v_G be the lexicographically smallest point x in the polytope $P_G := \bigcap_{h \in G} h$ (i.e. the result of the intersection of the half spaces in G) that minimizes $f(x)$. The condition of working with lexicographically smallest vectors is analogous to the technique in the simplex methods for resolving degenerated cases so that the simplex method is guaranteed to terminate. If $P_G \neq \emptyset$ and $H_+ \subseteq G$, then due to $c > 0$ there always exists such a v_G. If P_G is the empty set, then we define $v_G := \infty^d$, which represents infeasibility. If the solution is unbounded, then we define $v_G := -\infty^d$. We write $v_F < v_G$ if $f(v_F) < f(v_G)$ or if $f(v_F) = f(v_G)$ and v_F is lexicographically smaller than v_G. For an arbitrary set of constraints H we define $v^+_H = v_{H \cup H_+}$.

A set of constraints (half spaces) $B \subseteq G$ is called a *basis* of G if v_B is finite and $v_{B'} < v_B$ for every proper subset B' of B. For instance, H_+ is a basis of $H \cup H_+$ since $v_{H_+} = (0, 0, \ldots, 0)^T$ and the deletion of one of the constraints from H_+ would cause one coordinate to go to $-\infty$. Since $(0, 0, \ldots, 0)^T$ may not necessarily satisfy the constraints in $H \cup H_+$, the optimal solution v_B of a basis B may be infeasible for G, i.e., it may lie outside of the polytope P_G of G. A constraint $h \in H \cup H_+$ is *violated* by G iff $v_G < v_{G \cup \{h\}}$. The uniqueness of v_G implies that in this case $v_G \notin h$ (if v_G is finite). h is *extreme* in G if $v_{G-\{h\}} < v_G$, i.e., if h is violated by $G - \{h\}$.

With this terminology we can formulate the following lemma. Items 6.1 and 6.1 are obvious and item 6.1 is standard in the theory of linear optimization.

Lemma 6.1

(1) For each $F \subseteq G \subseteq H \cup H_+$ it holds $v_F \leq v_G$.

(2) If v_F and v_G are finite with $v_F = v_G$, then h is violated by F iff h is violated by G.

(3) If v_G is finite, then each basis of G has exactly d constraints and G has at most d extreme constraints.

6.2 A subexponential algorithm

First, we present a subexponential method called SUBEX_lp in order to search for an optimal solution of a linear optimization problem. SUBEX_lp receives as input a set G of m constraints and a basis $B \subseteq G$ of G and returns a tuple (v, B') with $v = v_G$ and a basis $B' \subseteq G$ such that $v_{B'} = v$. Thus, in order to obtain an optimal solution for $H \cup H_+$ we call SUBEX_lp($H \cup H_+, H_+$). SUBEX_lp uses a function Basis(B, h) which, given a basis B and a half space h that violates B, constructs a basis $B' \subseteq B \cup \{h\}$ for which B' does not violate h. I.e., B' contains h and omits a constraint from B. Using an appropriate representation of B, this task can be solved with a dual pivot step using $O(d^2)$ arithmetic operations. The violation test $v_B \notin h$ only needs $O(d)$ arithmetic operations.

A simple inductive proof (which is left as an exercise) shows that the algorithm SUBEX_lp always returns the correct output, if it terminates. Furthermore SUBEX_lp always terminates, since in every recursive call either G is shrunk or a basis B'' is chosen such that $v_{B''} > v_B$. Both G and the number of bases of G are finite such that at some point no further call of SUBEX_lp occurs.

Algorithm SUBEX_lp(G, B):
if $G = B$ **then**
  return (v_B, B)
else
  choose a random $h \in G - B$
  $(v, B') :=$SUBEX_lp($G - \{h\}, B$)
  if B' violates h **then**
    return SUBEX_lp(G,Basis(B', h))
  else
    return (v, B')

Figure 1: The SUBEX_lp algorithm.

In order to determine the expected runtime of SUBEX_lp we first consider the probability that a recursive call with Basis(B', h) is made. This is only the case if $v_G > v_{G-\{h\}}$, i.e., an extreme constraint h is removed from G. By Lemma 6.1 we know that at most d extreme constraints are contained in $G = H \cup H_+$. If $d - j$ of these constraints are contained in B, then the probability of choosing one out of the remaining j extreme constraints from $G - B$ at most $j/(m-d)$ with $|G| = m$. It remains to show that j quickly converges against 0 in the recursive calls. For this we enumerate the constraints in G such that

$$v_{G-\{h_1\}} \leq v_{G-\{h_2\}} \leq \ldots \leq v_{G-\{h_{d-k}\}} < v_B \leq v_{G-\{h_{d-k+1}\}} \leq \ldots \leq v_{G-\{h_m\}}$$

This order is not unique but the parameter k is, and it is called the *hidden dimension* of the pair (G, B). Note that for all $h \in G - B$ we have $v_B \leq v_{G-\{h\}}$ since $B \subseteq G - \{h\}$, so $h_1, h_2, \ldots, d_{d-k}$ are in B. Also, these must be extreme since by our assumption above, $v_{G-h_i} < v_B \leq v_G$ for all $1 \leq i \leq d - k$. Suppose now that an $h = h_i$ is chosen with $i > d - k$. Then for basis B' computed in the recursive call, $v_{B'} = v_{G-h}$, so the only way that B' violates h is that h is extreme. Hence, the only choices for h which can cause a second call are $h_{d-k+1}, \ldots, h_d$. Assume that h_{d-k+i} with $1 \leq i \leq k$ is chosen and SUBEX_lp($G - \{h\}, B$) returns a basis B' with $v_{B'} = v_{G-\{h_{d-k+i}\}}$. Then we compute $B'' =$Basis(B', h_{d-k+i}). Since $v_{G-\{h_{d-k+i}\}} = v_{B'} < v_{B''}$ the pair (G, B'') has a hidden dimension of at most $k - i$. Once the hidden dimension is 0, all extreme half spaces are in B, which means that $v_B = v_G$.

The hidden dimension is monotonic, i.e., if $B \subseteq F \subseteq G$, then the hidden dimension of (F, B) is not bigger than the hidden dimension of (G, B). This holds since the constraints $h_1, h_2, \ldots, h_{d-k}$ are contained in B (and thus also in F) and $v_{F-\{h_i\}} \leq v_{G-\{h_i\}}$ for all i, since $F \subseteq G$.

Let $b(m, k)$ be the worst case expected number of calls of Basis(B', h) by a call of SUBEX_lp(G, B) with m constraints and a hidden dimension of at most k. Then $b(d, k) = 0$ for all $0 \leq k \leq d$ and from the discussion from above we conclude

$$b(m, k) \leq b(m-1, k) + \frac{1}{m-d} \sum_{i=1}^{\min\{k, m-d\}} (1 + b(m, k-i)) \quad \text{for all } m > d.$$

With some technical effort one can show [2]

$$1 + b(m, k) \leq \exp\left(2\sqrt{k \,\underline{\ln}\frac{m-d}{\sqrt{k}}} + (\ln 3 + 2)\sqrt{k} + \underline{\ln}\frac{m-d}{\sqrt{k}}\right) = e^{O(\sqrt{k \ln(m-d)})}$$

where $\underline{\ln}\ x = \max\{\ln x, 1\}$. Each Basis call needs $O(d^2)$ arithmetic operations. For each basis B computed in the algorithm we check (due to the parameter $G - \{h\}$) in the recursive call each constraint h at most once for a violation by B. Thus, the number of violation tests is at most $(m - d)b(m, d)$ with $O(d)$ arithmetic operations in each test. Suppose we apply SUBEX_lp on an LP with $c \geq 0$ and constraint set H as described in Figure 2. Then, the following lemma is obtained.

Lemma 6.2 *For $n = |H|$ the algorithm SUBEX_lp(H) computes the vector $v_{H \cup H_+}$ with an expected number of $O((d^2 + nd)e^{O(\sqrt{d \ln n})})$ arithmetic operations.*

As we will see in the following, this bound can be improved.

Algorithm subex_lp(H):
(v, B) :=SUBEX_lp($H \cup H_+, H_+$)
return v

Figure 2: The subexponentiel LP algorithm.

6.3 Clarkson's Algorithm 2

The algorithm presented in this section uses a random sample R of $6d^2$ constraints and computes v_R^+ with the help of subex_lp(R). Then the violated constraints V of H are determined and their probability for being chosen the next time is increased. This is achieved by assigning each constraint $h \in H$ a multiplicity of μ_h. Initially μ_h equals 1, and it is doubled each time h is violated. The analysis will show that for any basis B of H the multiplicity of B increases so quickly that it is chosen after at most a logarithmic number of iterations. The details of this algorithm can be found in Figure 3. Here, we consider $H(\mu)$ as a multiset where each $h \in H$ occurs $\mu - h$ times in $H(\mu)$. Let $\mu(H) = \sum_{h \in H} \mu_h$.

Algorithm Clarkson2(H):
if $|H| \leq 6d^2$ **then**
    return subex_lp(H)
else
    $r := 6d^2$
    for all $h \in H$ **do** $\mu_h := 1$
    repeat
        choose a random multiset R of size r from $H(\mu)$
        v :=subex_lp(R)
        $V := \{h \in H(\mu) \mid v \text{ violates } h\}$
        if $|V| \leq \frac{1}{3d}|H(\mu)|$ **then**
            for all $h \in V$ **do** $\mu_h := 2\mu_h$
    until $V = \emptyset$
    return v

Figure 3: Clarkson's algorithm 2.

Note that $H(\mu)$ is a multiset, so V can be a multiset as well. In order to determine the runtime of Clarkson2 we need a bound for the expected size of V. We will formulate the lemma in a more general way than necessary since we will need it again later. In our case we have $G = \emptyset$.

Lemma 6.3 *Let H be a multiset of n constraints over d variables and let $G \subseteq H$. For each $1 \leq r < n$ the expected size of $V_R = \{h \in H \mid v_{G \cup R}^+$ violates $h\}$ with for a random multiset R of size r from H is at most $d \cdot \frac{n-r}{r+1}$.*

Proof. Let $\binom{H}{r}$ be the space over all multisets of r elements in H, i.e., all results for R. By definition of the expected value it holds

$$\mathbb{E}[|V_R|] = \frac{1}{\binom{n}{r}} \sum_{R \in \binom{H}{r}} |V_R|$$

For $R \in \binom{H}{r}$ and $h \in H$ let $X_G(R, h)$ be the indicator variable for the event that $v_{G \cup R}^+$ violates the constraint h. Then we

have

$$\begin{aligned}\binom{n}{r}\mathbb{E}[|V_R|] &= \sum_{R\in\binom{H}{r}} |V_R| = \sum_{R\in\binom{H}{r}} \sum_{h\in H-R} X_G(R,h)\\ &\overset{(1)}{=} \sum_{Q\in\binom{H}{r+1}} \sum_{h\in Q} X_G(Q-h,h)\\ &\overset{(2)}{\leq} \sum_{Q\in\binom{H}{r+1}} d = \binom{n}{r+1}\cdot d\end{aligned}$$

Here, equation (1) is deduced from the fact that choosing a set R of r constraints from H and subsequently choosing $h \in H - R$ is the same as choosing a set Q of $r+1$ constraints from H and the subsequent removal of h from Q. Equation (2) follows from the fact that $X_G(Q-h,h)$ is 1 only if $v_{G\cup Q\cup H_+-\{h\}} < v_{G\cup Q\cup H_+}$, i.e., if h is an extreme constraint in $G \cup Q \cup H_+$, and from the fact that by Lemma 6.1 there are at most d extreme constraints in each multiset of constraints. Thus,

$$\mathbb{E}[|V_R|] \leq d \cdot \frac{\binom{n}{r+1}}{\binom{n}{r}} \leq d \cdot \frac{n-r}{r+1}$$

□

For $r = 6d^2$ it holds that $d \cdot \frac{n-r}{r+1} \leq n/(6d)$. From this lemma and the Markov inequality it follows that the probability for $|V| > \frac{1}{3d}|H(\mu)|$ in Clarkson2 is at most $1/2$ and thus the expected number of loop iterations until a *successful* iteration is reached, i.e., $|V| \leq \frac{1}{3d}|H(\mu)|$, is at most 2. Furthermore, the following lemma regarding the size of an optimal basis holds.

Lemma 6.4 *Let $k \in \mathbb{N}$ and B an arbitrary optimal basis for $H \cup H_+$ (i.e. $v_B^+ = v_H^+$). After $k \cdot d$ successful iterations,*

$$2^k \leq \mu(B) < n \cdot e^{k/3}.$$

Proof. Each successful iteration increases the multiplicity of H by a factor of at most $(1 + 1/(3d))$. Therefore,

$$\mu(B) \leq \mu(H) \leq n(1 + 1/(3d))^{k\cdot d} < n \cdot e^{k/3}$$

It remains to determine the upper bound. For each successful iteration with $V \neq \emptyset$ it must hold: $v_R < v_{H\cup H_+} = v_B$. Thus, at least one constraint in B must be violated. Hence, in $k \cdot d$ successful iterations a constraint in B is doubled at least $k \cdot d$ times. Since B contains exactly d constraints, there must exist a constraint in B that is doubled at least k times, i.e., $\mu(B) \geq 2^k$. □

Finally, we need the following statement.

Lemma 6.5 *For $n = |H| > 6d^2$ the algorithm Clarkson2 computes the value v_H^+ in expected $O(d^2 n \log n)$ arithmetic operations and an expected number of at most $6d \ln n$ calls of SUBEX_lp with at most $6d^2$ constraints.*

Proof. For $k = 3 \ln n$ it holds: $2^k = 2^{3(\log n)/(\log e)} = n^{3/\log e} > n^2 = n \cdot e^{k/3}$. Thus, by Lemma 6.4 there are at most $3d \ln n$ successful iterations and by Lemma 6.3 the expected number of all iterations is at most $6d \ln n$. In each iteration at most $O(dn)$ arithmetic operations for the choice of R and the computation of V are needed and SUBEX_lp is called once. □

Thus, we obtain the following theorem.

Theorem 6.6 *For $n = |H|$, Clarkson2(H) computes $v_{H\cup H+}$ with an expected number $O(d^2 n \log n + e^{O(\sqrt{d \ln d})} \log n)$ of arithmetic operations.*

But this is not the best we can do. A further improvement is achieved by the algorithm presented in the next section.

6.4 Clarkson's Algrithm 1

In Clarkson's algorithm 1, for a set H of n constraints we choose a random multiset R of size $d\sqrt{n}$, compute $v = v_R^+$, and determine the set V of constraints in H that are violated by v. If $|V| \leq 2\sqrt{n}$ then we add V to an initially empty set G, choose a further random set R, compute $v_{G\cup R}^+$, add the violated constraints to G, and so on, until there is no more violated constraints and we can therefore determine a solution. Details of this method can be found in Figure 4.

Algorithm Clarkson1(H):
if $|H| \leq 9d^2$ **then**
    return Clarkson2(H)
else
    $r := \lfloor d\sqrt{n} \rfloor$; $G := \emptyset$
    repeat
        choose a random set R of size r from H
        $v :=$Clarkson2($G \cup R$)
        $V := \{h \in H \mid v \text{ violates } h\}$
        if $|V| \leq 2\sqrt{n}$ **then** $G := G \cup V$
    until $V = \emptyset$
    return v

Figure 4: Clarkson's Algorithm 1.

Lemma 6.3 implies that for $r = \lfloor d\sqrt{n} \rfloor$ it holds: $\mathbb{E}[|V|] \leq d \cdot \frac{n-r}{r+1} \leq \sqrt{n}$. By the Markov inequality the probability for $|V| > 2\sqrt{n}$ is at most $1/2$ and thus the expected number of loop iterations until $|V| \leq 2\sqrt{n}$ is at most 2. Furthermore, it holds that if any constraint is violated by $v = v_{G\cup R}^+$, then for any optimal basis B of $H \cup H_+$ (i.e. $v_B^+ = v_H^+$) there must exist a constraint $h \in B - (G \cup R \cup H_+)$ that is violated by v. (If no constraint from B is violated by v, then $v_H^+ = v_B^+ \leq v_{G\cup R\cup B}^+ = v_{G\cup R}^+ \leq v_H^+$, and therefore v must be a solution of H.) Since this constraint is added to G, G can be augmented at most d times. Thus, we get:

Lemma 6.7 *For $n = |H| > 9d^2$ the algorithm Clarkson1 computes v_H^+ in an expected number $O(d^2 n)$ of arithmetic operations and an expected number of at most $2d$ calls of Clarkson2 with at most $2d\sqrt{n}$ constraints.*

From this lemma and Theorem 6.6 the following theorem is obtained.

Theorem 6.8 *For $n = |H|$, Clarkson1(H) computes the vector $v_{H\cup H+}$ with an expected number of $O(d^2 n + d^4\sqrt{n}\log n + e^{O(\sqrt{d \ln d})}\log n)$ arithmetic operations.*

This implies the following result.

Corollary 6.9 *A linear program with a constant number of variables can be solved in linear time in the number of constraints.*

For the case of a fixed number of variables we do not need to apply the complex SUBEX_lp algorithm to obtain a linear run time. In this case it suffices to enumerate all corners of the polytope defined by H (which are at most $2^{|H|}$, i.e., a constant number) instead of calling SUBEX_lp(H) for $|H| \leq 6d^2$.

6.5 Abstract optimization problems

In this section we will generalize Clarkson's algorithm to so-called abstract optimization problems. This includes, for instance, the problem of the smallest enclosing circle, which we already presented in the first chapter.

An *abstract optimization problem* is a tuple (H, v) where H is a finite set and $v : 2^H \to T$ is a function that maps subsets from H to values in an ordered set $(T, \leq)$. The set T contains a maximal element ∞. We require two axioms:

- **Monotonicity:** For all F, G with $F \subseteq G \subseteq H$ it holds: $v(F) \leq v(G)$.

- **Locality:** For all F, G with $F \subseteq G \subseteq H$ and $v(F) = v(G) < \infty$ it holds: each $h \in H$ with $v(G) < v(G \cup \{h\})$ implies $v(F) < v(F \cup \{h\})$.

Abstract optimization problems that fulfill both axioms are of *LP type*.

Let $v(H) < \infty$. A minimal subset $B \subseteq H$ with $v(B') < v(B)$ for all proper subsets B' of B is called a *basis* of H. An *optimal basis* is a basis B with $v(B) = v(H)$. The maximum cardinality of a basis is called the *combinatorial dimension* of (H, v) and is denoted by $dim(H, v)$. We will consider several examples in order to illustrate these definitions.

Linear optimization

In this case, H is the set of all linear constraints and $v(H)$ denotes the optimum value in the polytope for H with respect to the given objective function. The monotonicity condition obviously holds in this case. Also the locality condition holds since if $v(G) < v(G \cup \{h\})$ (i.e., h is violated by v_G), then due to $F \subseteq G$, h is also violated by v_F. The combinatorial dimension is simply the number of variables of the LP.

Smallest enclosing circle

In this case, H is the set of points and $v(H)$ denotes the radius of the smallest enclosing circle for H. The monotonicity condition can be verified easily. Also the locality condition holds since if the smallest enclosing circle for G and $F \subseteq G$ are of the same size (and thus they actually are the same circles when lexicographically ordering the solutions) and point h lies outside of the circle of G, then h must also lie outside of the circle of F. Since in the 2-dimensional case at most 3 points are sufficient for determine the smallest enclosing circle for H, the combinatorial dimension of this problem is 3. For d dimensions, at most $d + 1$ points are sufficient. Thus, the following theorem holds.

Theorem 6.10 *The problem of the smallest enclosing circle for n points in a d-dimensional Euclidean space for a constant d can be solved in time $O(n)$.*

Polytope distance

Given two polytopes P and Q which are specified by their corner sets V_P and V_Q (and are given as the convex hull of these), compute the points $p \in P$ and $q \in Q$ with minimum distance, which we denote by $dist(V_p, V_Q)$. For this problem, $H = (U, V)$ for two point sets U and V and $G = (U', V') \subseteq H$ iff $U' \subseteq U$ and $V' \subseteq V$. Furthermore, we define $v(H) = -dist(U, V)$. Again, the monotonicity condition can easily be verified. Also the locality condition holds since if the distances for G and $F \subseteq G$ are equal (and thus the witnesses for them in F and G are equal when lexicographically ordering the solutions according to them) and point h decreases the distance for G, then h must also decrease the distance for F. (A formal proof is left as an exercise.) In the 2-dimensional case, at most 3 points are sufficient (one point in U and two points in V which specify an edge of the polytope for V, or the other way round) for computing the minimum distance between the polytopes specified by U and V. In the d-dimensional case, at most $d + 1$ points are sufficient (one point in U and d points in V). The combinatorial dimension of this problem is $d + 1$. Thus it holds:

Theorem 6.11 *The polytope distance problem in the d-dimensional Euclidean space for a constant d can be solved in time $O(n)$.*

6.6 Clarkson's algorithms for LP type problems

The Clarkson1 algorithm can be generalized to LP type problems as shown in Figure 5.

Here, B is an optimal basis of the problem restricted to $G \cup R$ and h is violated by B if $v(B) < v(B \cup \{h\})$. Lemma 6.3 can be generalized as follows.

Lemma 6.12 *Let (H, v) be an abstract optimization problem of LP type with $|H| = n$. Let H' be a multiset of H over d variables and let $G \subseteq H'$. For each $1 \leq r < n$ it holds that the expected size of $V_Q = \{h \in H \mid v_Q^+ \text{ violates } h\}$ with $Q = G \cup R$ for a random multiset R of size r from H is at most $dim(H, v) \cdot \frac{n-r}{r+1}$.*

The proof is analogous to Lemma 6.3. Also the Clarkson2 algorithm can be modified as shown in Figure 6. Finally, for the SUBEX_lp algorithm there is an abstract formulation as shown in Figure 7.

Algorithm Clarkson1(H):
if $|H| \leq 9dim(H,v)^2$ **then**
    return Clarkson2(H)
else
    $r := \lfloor dim(H,v)\sqrt{n} \rfloor$; $G := \emptyset$
    repeat
        choose a random set R of size r from H
        $B :=$Clarkson2($G \cup R$)
        $V := \{h \in H \mid B \text{ violates } h\}$
        if $|V| \leq 2\sqrt{n}$ **then** $G := G \cup V$
    until $V = \emptyset$
    return $v(B)$

Figure 5: Clarkson's Algorithm 1.

Algorithm Clarkson2(H):
if $|H| \leq 6dim(H,v)^2$ **then**
    return subex_lp(H)
else
    $r := 6dim(H,v)^2$
    for all $h \in H$ **do** $\mu_h := 1$
    repeat
        choose a random multiset R of size r from $H(\mu)$
        $B :=$subex_lp(R)
        $V := \{h \in H(\mu) \mid B \text{ violates } h\}$
        if $|V| \leq \frac{1}{3dim(H,v)}|H(\mu)|$ **then**
            for all $h \in V$ **do** $\mu_h := 2\mu_h$
    until $V = \emptyset$
    return B

Figure 6: Clarkson's Algorithm 2.

Algorithm SUBEX_lp(H):
$B' :=$SUBEX_lp(H, B) for a basis B of H
return B'

Algorithm SUBEX_lp(G, B):
if $G = B$ **then**
    return B
else
    choose a random $h \in G - B$
    $B' :=$SUBEX_lp($G - \{h\}, B$)
    if B' violates h **then**
        return SUBEX_lp(G,Basis(B', h))
    else
        return B'

Figure 7: The subexponential LP algorithm.

Therefore, one only has to think about how to implement the basis algorithm and how to check that B violates an element h. For the smallest enclosing circle example, for Basis(B', h) a smallest enclosing circle which contains the points in B' and h has to be computed, and for the violation test one has to check whether the point h lies outside of the circle specified by B.

References

[1] K.L. Clarkson. Las Vegas algorithms for linear and integer programming when the dimension is small. *Journal of the ACM*, 42:448–499, 1995.

[2] B. Gärtner. *Randomized Optimization by simplex-type methods*. PhD thesis, Freie Universität Berlin, 1995.

[3] B. Gärtner und E. Welzl. Linear programming – randomization and abstract frameworks. In *Proc. of the Symp. on Theoretical Aspects of Computer Science (STACS)*, pages 669–687, 1996.