# Explicit and Implicit Enforcing – Randomized Optimization

Bernd Gärtner and Emo Welzl

Theoretical Computer Science, ETH Zürich, CH-8092 Zürich, Switzerland

## 1 Introduction

This tutorial is aiming at some randomized approaches for geometric optimization that have been developed over the last ten to fifteen years. The methods are simple, sometimes even the analysis is. The simplicity of the methods entails that they use very little of the underlying structure of the problems to tackle, and, as a consequence, they are applicable to a whole range of problems. Most prominently, they have led to new bounds for combinatorial solutions to linear programming. But with little adaption necessary – basically few problem specific primitive operations have to be supplied – the methods can be used for computing smallest enclosing balls of point sets, smallest volume enclosing ellipsoids, the distance between polytopes (given either as convex hulls or as intersection of halfspaces), and for several other problems. If the dimension of the input instance is constant, then several linear time methods are available, and some of them even behave 'well' when the dimension grows; at least, no better provable behavior is known. Many of the methods can be interpreted as variants of the simplex algorithm, with appropriately chosen randomized pivot rules.

After the good news, the even better news. It seems that there is ample space for improvement. The fact that the methods are so general is nice, but we would be more than happy to use more of the structure, if we only knew how. The ultimate goal is to derive a polynomial combinatorial linear programming algorithm.

*Disclaimer.* We are not aiming at the most succinct presentation of the most efficient procedures (let it be theoretically or practically). Instead, we take a stroll around the topic with emphasis on the methods, which we believe to be of interest in their own right. And we enjoy how binomial coefficients and cycle numbers enter the picture. Usefulness is definitely not our primary concern!

Now that we got rid of the utilitarian fanatic, let us disclose that *some* of the ideas discussed here have found their way into efficient implementations (see, e.g., [6]).

For the rest of this section we briefly discuss two processes similar in structure to what we will later use for computing smallest enclosing balls, the problem of reference for this presentation. This algorithmic problem was posed by James Joseph Sylvester [18] in 1857. In Sect. 2 we settle basic terminology and properties needed for the algorithmic treatment of that problem, and we analyze a first process that computes a smallest enclosing ball of a finite point set. Next, in Sect. 3, we present the first two efficient procedures: One is linear in $n$, the number of points, but has exponential behavior in terms of the dimension $d$; the other one is exponential in $n$, but it is faster than the first one when the dimension is large ($d \approx n$). Assuming a fast solution for $n = 2(d+1)$, Sect. 4 presents a subexponential process. Discussion and bibliographical remarks conclude the paper.

We are interested in the processes, emphasizing their common and their distinguished features. We omit a presentation of the best known method for solving the problem when $n \approx d$, which is quite technical, as it stands now. The best known bounds are summarized in the discussion.

*Minimum Finding Procedures.* Let us start with two simple procedures for computing the minimum of a set of real numbers. These procedures may seem somewhat stupid, and, in fact, they were, if we were really interested in computing minima. However, we are interested in the processes they define, and in their analyses, not in the output they produce. They already provide schemes which are able to solve more difficult problems. Here is a first try. We assume that we are given a set $A$ of $n$ real numbers. The procedure AlwaysCheckMin performs

$A \subseteq \mathbf{R}$, finite.
PRECONDITION:
None.
POSTCONDITION:
$x = \min A$.

```
function AlwaysCheckMin(A)
if A = ∅ then
    x ← ∞;
else
    a ←random A;
    x ← AlwaysCheckMin(A \ {a});
    if a < x then
        x ← a;
        for all b ∈ A do
            if b < x then
                output "Something wrong!";
return x;
```

the normal scan through $A$ in random order, always substituting a current minimum if a smaller number is encountered. However, whenever a new minimum is encountered, we check it against all previous numbers (not trusting the transitivity of the $\leq$-relation[1]). We use here and further on the statement '$s \leftarrow_{\text{random}} S$;' for assigning to $s$ an element uniformly at random from set $S$. The expected number, $t(n)$, of comparisons ('$a < x$' or '$b < x$') of the procedure when $A$ has cardinality $n$ obeys the recursion

$$\textsf{AlwaysCheckMin}\text{-recursion, comparisons}$$
$$t(n) = \begin{cases} 0, & \text{if } n = 0, \\ t(n-1) + 1 + \frac{1}{n} \cdot n, & \text{if } n \geq 1, \end{cases}$$

since the randomly chosen element $a$ is the minimum of $A$ with probability[2] $\frac{1}{n}$. Hence, $t(n) = 2n$. If some of the numbers are equal, the expected number of comparisons decreases.

Here is a second procedure, called TryAndTryMin, that computes the minimum of a nonempty set $A$ of real numbers. Note that the function invokes a

| | |
|---|---|
| $A \subseteq \mathbf{R}$, finite.<br>PRECONDITION:<br>$A \neq \emptyset$.<br>POSTCONDITION:<br>$x = \min A$. | **function** TryAndTryMin($A$)<br>$a \leftarrow_{\text{random}} A$;<br>**if** $A = \{a\}$ **then**<br>    $x \leftarrow a$;<br>**else**<br>    $x \leftarrow$ TryAndTryMin($A \setminus \{a\}$);<br>    **if** $a < x$ **then**<br>        $x \leftarrow$ TryAndTryMin($A$);<br>**return** $x$; |

recursive call with the same arguments – something that is prohibited in a deterministic procedure that we wish to eventually terminate. The strange feature of the procedure is, of course, that it ignores the fact

$$a < \min(A \setminus \{a\}) \quad \Longrightarrow \quad a = \min A \;.$$

It will be our task to find corresponding facts in some of the higher-dimensional counterparts, when they are not as self-evident as here.

[1] Perhaps its not so ridiculous at all. Just imagine the numbers are given implicitly, as $x$-coordinates of intersections of lines, say. Then numerical problems in calculations and comparisons may quite easily lead you into contradictions. But, again, that's not the point here!

[2] If not all numbers are distinct (i.e. $A$ is a multiset), the probability is *at most* $\frac{1}{n}$.

For $n \in \mathbf{N}$, the expected number, $t(n)$, of comparisons of the procedure when applied to a set of $n$ elements satisfies the recursion

$$t(n) = \begin{cases} 0, & \text{if } n = 1, \\ t(n-1) + 1 + \frac{1}{n} \cdot t(n), & \text{otherwise,} \end{cases}$$

TryAndTryMin-recursion, comparisons

since the second recursive call happens iff $a = \min A$, which happens with probability $\frac{1}{n}$. For $n > 1$, this can be rewritten as (by moving the $t(n)$-term from the right to the left side, and dividing by $(n-1)$)

$$\frac{t(n)}{n} = \frac{t(n-1)}{n-1} + \frac{1}{n-1} = \frac{1}{n-1} + \frac{1}{n-2} + \cdots + \frac{1}{1} + \frac{t(1)}{1}$$

which yields

$$\frac{t(n)}{n} = H_{n-1} \quad \Longrightarrow \quad t(n) = nH_{n-1} \ ,$$

where $H_m$, $m \in \mathbf{N}_0$, is the $m$th *Harmonic number* $H_m := \sum_{i=1}^{m} \frac{1}{i}$; $\ln(m+1) \leq H_m \leq 1 + \ln m$ for $m \in \mathbf{N}$.

If $A$ is a multiset, the recursion still holds, but now with inequality (the second recursive call happens iff $a$ is the unique minimum in $A$); the expected number of steps can only decrease.

*Notation.* $\mathbf{N}$ denotes the positive integers, $\mathbf{N}_0$ the nonnegative integers, $\mathbf{R}$ the real numbers, $\mathbf{R}^+$ the positive reals, and $\mathbf{R}_0^+$ the nonnegative reals.

$k \in \mathbf{N}_0$; $A$ a finite set. $\#A$ denotes the cardinality of $A$, $\binom{A}{k}$ the family of $k$-element subsets of $A$, and $2^A$ the family of all subsets of $A$.

$n, k \in \mathbf{N}_0$. $n^{\underline{k}}$ denotes the *falling power* $\prod_{i=0}^{k-1}(n-i)$, $\binom{n}{k}$ the *binomial coefficient* $\frac{n^{\underline{k}}}{k!}$, and $\left[{n \atop k}\right]$ the *cycle number* (Stirling number of the first kind). It counts the number of permutations of $n$ elements with $k$ cycles. Equivalently, it is defined by $\left[{0 \atop 0}\right] = 1$, and for $n, k \in \mathbf{N}$, $\left[{n \atop 0}\right] = \left[{0 \atop k}\right] = 0$ and

$$\left[{n \atop k}\right] = (n-1)\left[{n-1 \atop k}\right] + \left[{n-1 \atop k-1}\right] .$$

For properties of falling powers, binomial coefficients and cycle numbers the reader is invited to consult the exposition [11].

$d \in \mathbf{N}$; $x \in \mathbf{R}^d$. $\|x\|$ denotes the length of vector $x$ (2-norm).

$A \subseteq \mathbf{R}$. We use $\max A$, $\sup A$, $\min A$, and $\inf A$ to denote the maximum, supremum, minimum, and infimum, resp., of $A$. As usual, $\min \emptyset = \inf \emptyset = \infty$ and $\max \emptyset = \sup \emptyset = -\infty$.

*Convention.* Throughout this text, $d$ is a positive integer (the dimension of the ambient space), and $B, C, E, F, P, S \subseteq \mathbf{R}^d$ are *finite* without further explicit mention.

## 2 Smallest Enclosing Ball

Here is our reference problem for this presentation. We are given a set $P$ of points in $\mathbf{R}^d$ and we are to compute a closed ball of smallest radius containing $P$. The nice feature of this optimization problem (as opposed to linear programming, say) is that a solution always exists, and it is always unique.

$z \in \mathbf{R}^d$; $r \in \mathbf{R}$. We use $b_{z,r}$ for the closed ball with center $z$ and radius $r$, $b_{z,r} := \{p \in \mathbf{R}^d \mid \|p - z\| \le r\}$. Given such a ball $b$, we let $\partial b$ denote the boundary of $b$. If $b = b_{z,r} \ne \emptyset$, we let $z_b := z$ and $r_b := r$; $r_\emptyset := -\infty$.

Note that $b_{z,r}$ degenerates to a point iff $r = 0$, and to the empty set iff $r < 0$.

**Definition 1 (smallest enclosing radius).** $P \subseteq \mathbf{R}^d$.

$$\rho P := \inf_{\text{closed ball } b \supseteq P} r_b \ .$$

In particular, $\rho\emptyset = -\infty$; $\rho P = 0$ iff $\#P = 1$; and $\rho P > 0$, otherwise.

We omit the proof for the geometric lemma below. The reader may wish to read (2-4) as 'axioms' that are necessary for the later conclusions. In this way the generality of the results becomes evident.

**Lemma 1.** $P \subseteq \mathbf{R}^d$.

(**1**) *There is a unique closed ball of radius $\rho P$ that contains $P$.*

(**2**) *If $P' \subseteq P$ then $\rho P' \le \rho P$.*

(**3**) *Let $P' \subseteq P$. Then $\rho P' = \rho P$ iff $\rho(P' \cup \{p\}) = \rho P'$ for all $p \in P \setminus P'$.*

(**4**) *If $\rho P' \ne \rho P$ for all proper subsets $P'$ of $P$, then $\#P \le d + 1$.*

While (1) and (4) require some geometric reasoning, (2) is immediate from the definition of $\rho$, and (3) is immediate from (1) and (2). In order to properly appreciate[3] (3), note that it is possible[4] for $P' \subseteq P$ that $\rho P' \ne \rho P$ while $\rho P = \rho(P \setminus \{p\})$ for all $p \in P \setminus P'$.

**Definition 2 (smallest enclosing ball, basis, basis of, extreme).** $P \subseteq \mathbf{R}^d$.

(**1**) $\mathsf{minB}P$ *denotes the ball of radius $\rho P$ covering $P$.*

(**2**) *$P$ is called a* basis *if $\mathsf{minB}P' \ne \mathsf{minB}P$ for all proper subsets $P'$ of $P$.*

(**3**) *A basis $B \subseteq P$ is called* basis <u>of</u> $P$ *if $\mathsf{minB}B = \mathsf{minB}P$.*

(**4**) *A point $p \in P$ is called* extreme in $P$ *if $\mathsf{minB}(P \setminus \{p\}) \ne \mathsf{minB}P$.*

As it can be easily seen, there is always a basis *of* $P$. But, in general, the basis of a set $P$ is not unique[5]. (Take, for example, the four vertices of a square. Observe also, that this set has no extreme points at all.) Moreover, note that

$$\mathsf{minB}P' = \mathsf{minB}P \iff \rho P' = \rho P \ , \quad \text{if } P' \subseteq P.$$

[3] Another exercise for appreciation is to define $\sigma P$ as the smallest volume of a simplex containing $P$, and then to see how a statement analogous to Lemma 1(3) fails.

[4] This can, however, be excluded by certain general position assumptions.

[5] It is unique under appropriate general position assumptions.

**Lemma 2.** *$P \subseteq \mathbf{R}^d$.*

**(1)** *A point $p \in P$ is extreme in $P$ iff $p$ is contained in every basis of $P$.*

**(2)** *The number of extreme elements in $P$ is at most $d+1$.*

*Proof.* (1) Let $p$ be an extreme point in $P$ and let $B$ be a basis of $P$ which does not contain $p$. Then $\rho B \leq \rho(P \setminus \{p\}) < \rho P = \rho B$; a contradiction.

On the other hand, if $\mathsf{minB}P = \mathsf{minB}(P \setminus \{p\})$ – that is, $p$ is not extreme – then any basis of $P \setminus \{p\}$ is a basis of $P$ that does not contain $p$.

(2) Follows from (1) and Lemma 1(4). ◻

*Basic (Primitive) Operations.* Throughout the presentation we will measure algorithms in terms of some basic operations. The fact that we do not elaborate much on them does not imply that they are trivial to solve efficiently – definitely not so, when it comes to actual implementations. Two such basic operations for points in $\mathbf{R}^d$ are:

- *$(d,k)$-Basis computation* '$\mathsf{basis}^d_k\, S$', for $d, k \in \mathbf{N}$: Given a set $S \subseteq \mathbf{R}^d$ with $\#S \leq k$, it returns a basis of $S$.
- *Inclusion predicate* '$p \not\in B$': Given a *basis* $B$ and a point $p$, it decides whether $p$ is not in $\mathsf{minB}B$.

If an algorithm delivers a basis $B$ of the given point set $P$, then it is easy to determine $\mathsf{minB}B = \mathsf{minB}P$: compute center and radius of the unique ball in the affine hull of $B$ that has $B$ on its boundary[6].

If $\mathsf{basis}^d_k$ were available for all $k$, then, of course, there is little to do. However, we will assume that this operation is at disposal only for certain $k$'s dependent on $d$, ($k = d+1$ and $k = 2(d+1)$). Again, we do not claim that these are easy to solve – in fact, they are at the core of the problem, and no polynomial time solutions are known.

What is an obvious bound for obtaining a basis of a set $P$ of $n \geq d+1$ points in $\mathbf{R}^d$? We can go through all $(d+1)$-element subsets, apply $\mathsf{basis}^d_{d+1}$ to each of them, and check the obtained basis against all remaining $n-(d+1)$ points with inclusion tests. This requires $\binom{n}{d+1}$ calls to $\mathsf{basis}^d_{d+1}$ and $\binom{n}{d+1}(n-d-1)$ inclusion tests. Instead of checking each basis against all remaining points, we could simply search for a basis $B$ that maximizes $\rho B$ among all bases. The reader is encouraged to verify that this will indeed deliver a basis of $P$.

*Trial and Error.* In the spirit of the procedures in the previous section, and equipped with the basic operation $\mathsf{basis}^d_{d+1}$ and the predicate $p \not\in B$, we can easily provide a first randomized procedure called $\mathsf{TryAndTry}$ for our problem. It takes a set $S \subseteq \mathbf{R}^d$ and returns a basis of $S$. Correctness of the procedure is obvious, provided it eventually terminates.

[6] In time $O(d^3)$. Similarly, the inclusion tests can be implemented efficiently: If center and radius of $\mathsf{minB}B$ have been precomputed, in time $O(d)$; if not, in $O(d^3)$.

$S \subseteq \mathbf{R}^d$.

PRECONDITION:
None.

POSTCONDITION:
$B$ basis of $S$.

**function** $\mathsf{TryAndTry}(S)$<br>
**if** $\#S \leq d+1$ **then**<br>
  $B \leftarrow \mathsf{basis}^d_{d+1}\, S$;<br>
**else**<br>
  $p \leftarrow_{\text{random}} S$;<br>
  $B \leftarrow \mathsf{TryAndTry}(S \setminus \{p\})$;<br>
  **if** $p \notin B$ **then**<br>
    $B \leftarrow \mathsf{TryAndTry}(S)$;<br>
**return** $B$;

Let $t(n)$ denote the maximum[7] expected number of '$p \notin B$'-tests performed by the algorithm for an $n$-point set in $\mathbf{R}^d$. Then

$\mathsf{TryAndTry}$-recursion, inclusion tests

$$t(n) \begin{cases} = 0, & \text{if } n \leq d+1, \\ \leq t(n-1) + 1 + \frac{d+1}{n} \cdot t(n), & \text{otherwise,} \end{cases}$$

since there are at most $d+1$ extreme points in $S$ whose removal entails the second recursive call. Division of the inequality by $(n-1)^{\underline{d+1}}$ and carrying the $t(n)$-term to the left side gives[8]

$$\begin{aligned} \frac{t(n)}{n^{\underline{d+1}}} &\leq \frac{t(n-1)}{(n-1)^{\underline{d+1}}} + \frac{1}{(n-1)^{\underline{d+1}}} \\ &\leq \frac{1}{(n-1)^{\underline{d+1}}} + \frac{1}{(n-2)^{\underline{d+1}}} + \cdots + \frac{1}{(d+1)^{\underline{d+1}}} + \frac{t(d+1)}{(d+1)^{\underline{d+1}}} \\ &= \frac{1}{d}\left(\frac{1}{d^{\underline{d}}} - \frac{1}{(n-1)^{\underline{d}}}\right) \;=\; \frac{1}{(d+1)!}\left(1 + \frac{1}{d}\right) - \frac{1}{d(n-1)^{\underline{d}}} \end{aligned}$$

for $n > d+1$. The recursion for the maximum expected number, $b(n)$, of basis computations is

$\mathsf{TryAndTry}$-recursion, $\mathsf{basis}^d_{d+1}$-computations

$$b(n) \begin{cases} = 1, & \text{if } n \leq d+1, \\ \leq b(n-1) + \frac{d+1}{n} \cdot b(n), & \text{otherwise.} \end{cases}$$

For $n \geq d+1$, this matches exactly the recursion

$$\binom{n}{d+1} = \begin{cases} 1, & \text{if } n = d+1 \\ \binom{n-1}{d+1} + \frac{d+1}{n}\binom{n}{d+1}, & \text{if } n > d+1, \end{cases}$$

[7] By 'maximum expected' we mean the following: Let $T(S)$ be the expected number of tests for $S \subseteq \mathbf{R}^d$, and $t(n) := \sup_{\#S=n} T(S)$.

[8] For the identity for sums of reciprocals of falling powers consult [11, page 53, equation (2.53)].

for binomial coefficients.

**Theorem 1.** *$n \in \mathbf{N}_0$; $P \subseteq \mathbf{R}^d$, $\#P = n \geq d+1$.*
*Procedure* TryAndTry *computes a basis of $P$ with an expected number of at most*

$$\binom{n}{d+1}\left(1+\frac{1}{d}\right) - \frac{n}{d} \quad \textit{inclusion-tests}$$

*and with an expected number of at most*

$$\binom{n}{d+1} \quad \textit{employments of } \mathsf{basis}^d_{d+1}\textit{-computations.}$$

## 3 Boundary (Explicit) Enforcing

What is wrong with TryAndTry? Recall why we thought that TryAndTryMin from the previous section was so ridiculous: It ignored the useful information that $a < \min(A \setminus \{a\})$ implies that $a$ is already the minimum. Here things are not as immediate, but we know that $p \notin \mathsf{minB}(P \setminus \{p\})$ implies that $p$ is in every basis of $P$. Moreover, it can be shown that

$$p \notin \mathsf{minB}(P \setminus \{p\}) \quad \Longrightarrow \quad p \text{ lies on the boundary of } \mathsf{minB}P$$

We should transfer this useful information to the second recursive call, and we will do so by *explicitly forcing* $p$ on the boundary. If this extra information would suffice to allow a linear time solution, then the whole procedure would be linear, for reasons similar to the fact that function AlwaysCheckMin was linear.

Procedure Explicit (to be described shortly) computes the smallest enclosing ball of a given set $S$, with a given subset $E \subseteq S$ of points prescribed to lie on the boundary of the ball. We have to lay the basics for description and analysis first.

**Definition 3 (prescribed boundary: smallest enclosing radius).**
$E \subseteq P \subseteq \mathbf{R}^d$.

$$\overline{\rho}(P,E) := \inf_{\text{closed ball } b \supseteq P,\, \partial b \supseteq E} r_b \ .$$

In particular, if there is no ball that covers $P$ and has $E$ on its boundary then $\overline{\rho}(P,E) = \infty$.

**Lemma 3.** *$E \subseteq P \subseteq \mathbf{R}^d$.*

(**1**) *If $\overline{\rho}(P,E) \neq \infty$, there is a unique ball of radius $\overline{\rho}(P,E)$ that contains $P$ and has $E$ on its boundary.*

(**2**) *If $E' \subseteq E$ and $E' \subseteq P' \subseteq P$ then $\overline{\rho}(P',E') \leq \overline{\rho}(P,E)$.*

(**3**) *Let $E' \subseteq E$ and $E' \subseteq P' \subseteq P$. Then $\overline{\rho}(P',E') = \overline{\rho}(P,E)$ iff*

$$\overline{\rho}(P' \cup \{p\}, E') = \overline{\rho}(P',E') \quad \textit{for all} \quad p \in P \setminus P',$$

*and*

$$\overline{\rho}(P' \cup \{p\}, E' \cup \{p\}) = \overline{\rho}(P', E') \quad \textit{for all} \quad p \in E \setminus E'.$$

(**4**) *If* $\overline{\rho}(P', E') \neq \overline{\rho}(P, E) < \infty$ *for all* $(P', E') \neq (P, E)$ *with* $E' \subseteq E$ *and* $E' \subseteq P' \subseteq P$*, then* $\#P' \leq d + 1$.

Similar to Lemma 1, (2) is obvious from the definition, and (3) is a consequence of (1) and (2).

**Definition 4 (prescribed boundary: smallest enclosing ball, extreme).** $E \subset P \subseteq \mathbf{R}^d$.

(**1**) *If* $\overline{\rho}(P, E) < \infty$, $\mathsf{min}\overline{\mathsf{B}}(P, E)$ *denotes the ball of radius* $\overline{\rho}(P, E)$ *that contains* $P$ *and has* $E$ *on its boundary. If* $\overline{\rho}(P, E) = \infty$*, we set* $\mathsf{min}\overline{\mathsf{B}}(P, E) = \mathbf{R}^d$ *(indicating infeasibility).*

(**2**) *A point* $p \in P \setminus E$ *is called* extreme *in* $(P, E)$ *if* $\mathsf{min}\overline{\mathsf{B}}(P \setminus \{p\}, E) \neq \mathsf{min}\overline{\mathsf{B}}(P, E)$.

**Lemma 4.** $E \subseteq P \subseteq \mathbf{R}^d$.

(**1**) *If* $p \in P \setminus E$ *is extreme in* $(P, E)$*, then*

$$\mathsf{min}\overline{\mathsf{B}}(P, E) = \mathsf{min}\overline{\mathsf{B}}(P, E \cup \{p\})$$

*and*

$$\mathsf{min}\overline{\mathsf{B}}(P, E) = \mathbf{R}^d \quad \textit{or} \quad p \textit{ is affinely independent from the points in } E.$$

(**2**) *If* $E$ *is affinely independent and* $\mathsf{min}\overline{\mathsf{B}}(P, E) \neq \mathbf{R}^d$*, then the number of extreme points in* $(P, E)$ *is at most* $d + 1 - \#E$.

It is instructive to display the implication of assertion (1) of the lemma in the following form.

$$\mathsf{min}\overline{\mathsf{B}}(P, E) = \begin{cases} \mathsf{min}\overline{\mathsf{B}}(P \setminus \{p\}, E), & \text{if } p \in \mathsf{min}\overline{\mathsf{B}}(P \setminus \{p\}, E), \\ \mathsf{min}\overline{\mathsf{B}}(P, E \cup \{p\}), & \text{if } p \notin \mathsf{min}\overline{\mathsf{B}}(P \setminus \{p\}, E), \end{cases} \qquad (1)$$
$$\text{for all } p \in P \setminus E.$$

We will also need a new predicate[9]

- *(Boundary) inclusion predicate* '$p \notin F$': Given a set $F$ of affinely independent points and a point $p$, it decides whether $p$ is not in $\mathsf{min}\overline{\mathsf{B}}(F, F)$.

*Forcing Points on the Boundary.* Here is the procedure $\mathsf{Explicit}$ that computes the smallest enclosing ball of a set $S$ of points, with a set $E \subseteq S$ of points that are prescribed to lie on the boundary of the ball.

[9] The implementation of this predicate is in fact identical to that of the inclusion predicate for bases: We compute the unique ball in the affine hull of $F$ that has $F$ on its boundary. Its center and radius are also center and radius of $\mathsf{min}\overline{\mathsf{B}}(F, F)$. Now the predicate can be evaluated by a simple distance calculation.

$E, S \subseteq \mathbf{R}^d$.

PRECONDITION:
$E \subseteq S$; $E$ affinely independent;
$\mathsf{min}\overline{\mathsf{B}}(S,E) \neq \mathbf{R}^d$.

POSTCONDITION:
$E \subseteq F \subseteq S$; $F$ affinely independent;
$\mathsf{min}\overline{\mathsf{B}}(F,F) = \mathsf{min}\overline{\mathsf{B}}(S,E)$.

**function** $\mathsf{Explicit}(S, E)$
**if** $S = E$ **then**
  $F \leftarrow E$;
**else**
  $p \leftarrow_{\text{random}} S \setminus E$;
  $F \leftarrow \mathsf{Explicit}(S \setminus \{p\}, E)$;
  **if** $p \notin F$ **then**
    $F \leftarrow \mathsf{Explicit}(S, E \cup \{p\})$;
**return** $F$;

From the precondition and (1) it follows that the precondition is satisfied for the recursive calls and that the function returns a set $F \supseteq E$ of affinely independent points with $\mathsf{min}\overline{\mathsf{B}}(F,F) = \mathsf{min}\overline{\mathsf{B}}(S,E)$. Our original problem is solved by the call $\mathsf{Explicit}(P, \emptyset)$ – the arguments obviously satisfy the precondition.

For $k, n \in \mathbf{N}_0$, $t_k(n)$ counts the maximum expected number of inclusion tests when $\#(S \setminus E) = n$ and $d+1-\#E = k$. The parameter $k$ expresses the degrees of freedom[10]. Since there are at most $k$ extreme points in $S \setminus E$, we get the following recursion. Note that $k = 0$ means that $\#E = d+1$. Hence, any other point is affinely dependent on $E$, and so a recursive call with parameters $(S, E \cup \{p\})$ would lead to a contradiction. Therefore, $t_0(n) = t_0(n-1)+1$ for $n \in \mathbf{N}$.

Explicit-recursion, inclusion tests

$$t_k(n) \begin{cases} = 0, & \text{if } n = 0, \\ = t_0(n-1)+1, & \text{if } k = 0 \text{ and } n > 0, \\ \leq t_k(n-1) + 1 + \frac{k}{n} \cdot t_{k-1}(n-1), & \text{if } n \geq 1 \text{ and } k \geq 1. \end{cases}$$

From $t_0(n) = n$ we get $t_1(n) \leq 2n$, and that yields $t_2(n) \leq 5n$, etc. Hence, we might guess that there is a bound of the form $t_k(n) \leq c_k n$ for constants $c_k$, $k \in \mathbf{N}_0$. These constants satisfy

$$c_k = \begin{cases} 1, & \text{if } k = 0, \\ 1 + k c_{k-1}, & \text{if } k \geq 1. \end{cases}$$

Division by $k!$ lets us rewrite this in the form

$$\frac{c_k}{k!} = \frac{1}{k!} + \frac{c_{k-1}}{(k-1)!} = \frac{1}{k!} + \frac{1}{(k-1)!} + \cdots + \frac{1}{1!} + \frac{c_0}{0!},$$

and we conclude that

$$c_k = k! \sum_{i=0}^{k} \frac{1}{i!} = \begin{cases} 1, & \text{if } k = 0, \\ \lfloor \mathsf{e} k! \rfloor, & \text{if } k \geq 1. \end{cases}$$

[10] And note: Less freedom is better!

The inequality $t_k(n) \leq c_k n$ for all $k, n \in \mathbf{N}_0$ is now easy to prove by induction.

Let us also estimate the maximum expected number $b_k(n)$ of assignments '$F \leftarrow E$;' (when $S = E$). We call these the *base cases*[11], and we obtain the recursion

$$\text{Explicit-recursion, base cases}$$

$$b_k(n) \begin{cases} = 1, & \text{if } n = 0 \text{ or } k = 0, \\ \leq b_k(n-1) + \frac{k}{n} \cdot b_{k-1}(n-1), & \text{if } n \geq 1 \text{ and } k \geq 1. \end{cases}$$

We multiply both sides by $\frac{n!}{k!}$ to obtain

$$\overbrace{\frac{n!\, b_k(n)}{k!}}^{B_{k+1}(n+1):=} \leq n \cdot \underbrace{\frac{(n-1)!\, b_k(n-1)}{k!}}_{=B_{k+1}(n)} + \underbrace{\frac{(n-1)!\, b_{k-1}(n-1)}{(k-1)!}}_{=B_k(n)}$$

which reminds us of the recursion for the cycle numbers. In fact, we also have $B_1(n+1) = n! = \left[{n+1 \atop 1}\right]$ for $n \in \mathbf{N}_0$, but for $k \in \mathbf{N}$, $B_{k+1}(1) = \frac{1}{k!} \neq \left[{1 \atop k+1}\right] = 0$.

It can be shown that if '$\leq$' is replaced by '$=$' then the solution to the recurrence for $b_k(n)$ equals

$$\begin{aligned} \frac{1}{n!} \sum_{i=0}^{k} \binom{k}{i} i! \left[{n+1 \atop i+1}\right] &\leq \frac{1}{n!} \sum_{i=0}^{k} \binom{k}{i} i! \left( \frac{n!}{i!} (H_n)^i \right) \qquad \text{see Appendix, Lemma 10} \\ &= \sum_{i=0}^{k} \binom{k}{i} (H_n)^i \;=\; (1 + H_n)^k \end{aligned}$$

Unless $n$ is small compared to $d$, the number of base cases is much smaller than the number of inclusion tests, so precomputation pays off.

**Theorem 2.** *$n \in \mathbf{N}_0$; $P \subseteq \mathbf{R}^d$, $\#P = n$.*
*A call* $\mathsf{Explicit}(P, \emptyset)$ *computes a subset $F$ of $P$ with* $\mathsf{min}\overline{\mathsf{B}}(F,F) = \mathsf{minB}P$ *with an expected number of at most*

$$\lfloor \mathsf{e}(d+1)! \rfloor n \quad \textit{inclusion tests}$$

*and an expected number of at most*

$$\frac{1}{n!} \sum_{i=0}^{d+1} \binom{d+1}{i} \frac{1}{i!} \left[{n+1 \atop i+1}\right] \leq (1 + H_n)^{d+1} \quad \textit{base cases.}$$

[11] If you think about an actual implementation, then this is a relevant quantity, since it is (perhaps) advisable to precompute center and radius of $\mathsf{min}\overline{\mathsf{B}}(F,F)$ in $O(d^3)$, and return $F$ with this information, so that further inclusion tests can be performed more efficiently in $O(d)$.

*To Force or not to Force.* After choosing point $p$, procedure Explicit has two options, either (i) to omit the point $p$ from consideration, or (ii) to assume that $p$ is extreme and to force it on the boundary. And it decides to bet on (i) first, and look at (ii) only if (i) failed. This is a reasonable thing to do, since – most likely – a random point will not be extreme. But this is not true if the number of points is small compared to the dimension (or, more precisely, if the number of points available in $S \setminus E$ is small compared to the number $d + 1 - \#E$ of empty slots available). Actually, as you think about it, the recursion for $t_k(n)$ becomes somewhat strange when $n < k$: We account a 'probability' larger than 1 for proceeding to the second call (the recursion is still correct, but obviously not tight in that case).

So we design a new procedure ForceOrNot[12] which will perform favorably for $n$ small compared to $k$. This procedure chooses an arbitrary point $p$ in $S \setminus E$, and then tosses a coin to decide whether option (i) or (ii) as described above is pursued.

---

$E, S \subseteq \mathbf{R}^d$.

PRECONDITION:
$E \subseteq S$;
$S$ affinely independent.

POSTCONDITION:
$E \subseteq F \subseteq S$;
$\mathsf{min}\overline{\mathsf{B}}(F, F) = \mathsf{min}\overline{\mathsf{B}}(S, E)$.

```
function ForceOrNot(S, E)
if S = E then
    F ← E;
else
    p ←_some S \ E;
    x ←_random {0, 1};
    if x = 1 then
        F ← ForceOrNot(S \ {p}, E);
        if p ∉ (F) then
            F ← ForceOrNot(S, E ∪ {p});
    else
        F ← ForceOrNot(S, E ∪ {p});
        if p loose in F then
            F ← ForceOrNot(S \ {p}, E);
return F;
```

---

New features appear. First, if we force *some* point on the boundary we have no guarantee that a ball with the prescribed properties exists. We circumvent this problem by requiring that the points in $S$ are affinely independent. Hence, there exists a ball with $S$ on its boundary and so $\mathsf{min}\overline{\mathsf{B}}(S', E') \neq \mathbf{R}^d$ for all $E' \subseteq S' \subseteq S$. This implies that $\#S \leq d + 1$, which is not so much of a problem, since we wanted a procedure for small sets to begin with. Moreover, we make

[12] '$s \leftarrow_{\text{some}} S$;' assigns to $s$ an *arbitrary* element in $S$.

the following

General Position Assumption. (2)

*For all* $\emptyset \neq F \subseteq S$*, no point in* $S \setminus F$ *lies on the boundary of* $\mathsf{min}\overline{\mathsf{B}}(F, F)$.

Both assumptions, affine independence and general position, can be attained via symbolic perturbation techniques as long as the number of points does not exceed[13] $d + 1$.

Second, how do we know now that a call $\mathsf{ForceOrNot}(S, E \cup \{p\})$ delivers a description of the smallest ball enclosing $S$ and with $E$ on its boundary? We were not concerned about that before in procedure Explicit, since there we made such a call only if we knew that $p$ has to lie on the boundary of such a ball.

**Definition 5 (loose).** $F \subseteq \mathbf{R}^d$.
*A point* $p \in F$ *is called* loose in $F$ *if* $\mathsf{min}\overline{\mathsf{B}}(F, F \setminus \{p\}) \neq \mathsf{min}\overline{\mathsf{B}}(F, F)$.

The following lemmas show that we can check $\mathsf{min}\overline{\mathsf{B}}(F, F) = \mathsf{min}\overline{\mathsf{B}}(P, E)$ by local inclusion and looseness tests.

**Lemma 5.** $E \subseteq F \subseteq P \subseteq \mathbf{R}^d$, *$P$ affinely independent and in general position according to (2).*
$\mathsf{min}\overline{\mathsf{B}}(F, F) = \mathsf{min}\overline{\mathsf{B}}(P, E)$ *iff*

$$\mathsf{min}\overline{\mathsf{B}}(F \cup \{p\}, F) = \mathsf{min}\overline{\mathsf{B}}(F, F) \quad \textit{for all} \quad p \in P \setminus F \qquad (*)$$

*and*

$$\mathsf{min}\overline{\mathsf{B}}(F, F \setminus \{p\}) = \mathsf{min}\overline{\mathsf{B}}(F, F) \quad \textit{for all} \quad p \in F \setminus E. \qquad (**)$$

While (*) can be checked by the boundary inclusion predicate, we need a new predicate[14] corresponding to condition (**).

- *Looseness predicate* '$p$ loose in $F$': Given a set $F$ of affinely independent points and a point $p$, it decides whether $p$ is loose in $F$.

Now that the correctness of the procedure is established, let us consider efficiency. For a point $p$ in $S \setminus E$ we have that $\mathsf{min}\overline{\mathsf{B}}(S, E) = \mathsf{min}\overline{\mathsf{B}}(S, E \cup \{p\})$ or $\mathsf{min}\overline{\mathsf{B}}(S, E) = \mathsf{min}\overline{\mathsf{B}}(S \setminus \{p\}, E)$. Our general position assumption excludes that both are true. Hence, if we toss a coin to guess which one is true, then we are mistaken with probability $\frac{1}{2}$.

Let $t(n)$ denote the expected number of inclusion and looseness tests when $\#(S \setminus E) = n$. Then

ForceOrNot-recursion, inclusion and looseness tests

$$t(n) = \begin{cases} 0, & \text{if } n = 0, \\ t(n-1) + 1 + \frac{1}{2} \cdot t(n-1), & \text{otherwise.} \end{cases}$$

[13] And if the number $n$ of points exceeds $d + 1$, then embed $\mathbf{R}^d$ in $\mathbf{R}^{n-1}$, and perturb!

[14] An implementation of this predicate has to decide whether $p$ is in $\mathsf{min}\overline{\mathsf{B}}(F \setminus \{p\}, F \setminus \{p\})$, and, if so, whether the radius of this ball is smaller than that of $\mathsf{min}\overline{\mathsf{B}}(F, F)$. All of this can be done in time $O(d^3)$.

That is,

$$t(n) = 1 + \frac{3}{2} \cdot t(n-1) = 1 + \frac{3}{2} + \left(\frac{3}{2}\right)^2 + \cdots + \left(\frac{3}{2}\right)^{n-1} + \left(\frac{3}{2}\right)^n t(0)$$

and so $t(n) = 2\left(\left(\frac{3}{2}\right)^n - 1\right)$. For example, $t(10) \approx 113$, $t(20) \approx 6{,}684$, $t(30) \approx 383{,}500$, and $t(40) \approx 22{,}114{,}662$. Here is the recursion for the expected number of base cases.

$$\textsf{ForceOrNot}\text{-recursion, base cases}$$
$$b(n) = \begin{cases} 1, & \text{if } n = 0, \\ b(n-1) + \frac{1}{2} \cdot b(n-1), & \text{otherwise,} \end{cases}$$

which solves to $\left(\frac{3}{2}\right)^n$. We learnt that on the average roughly two inclusion tests are performed for each $F$ appearing in the procedure.

**Theorem 3.** *$n \in \mathbf{N}_0$; $P \subseteq \mathbf{R}^d$, $\#P = n$, $P$ affinely independent and in general position according to (2).*
*A call $\textsf{ForceOrNot}(P, \emptyset)$ computes a subset $F$ of $P$ with $\textsf{min}\overline{\textsf{B}}(F, F) = \textsf{minB}P$ with an expected number of*

$$2\left(\left(\frac{3}{2}\right)^n - 1\right) \quad \textit{inclusion and looseness tests}$$

*and an expected number of*

$$\left(\frac{3}{2}\right)^n \quad \textit{base cases.}$$

## 4 Implicit Enforcing

Throughout this section we will use $\delta$ short for $d + 1$.

The procedure of the section, Implicit, has a very similar look and feel as Explicit and TryAndTry. Besides the set $S$ under consideration, it has as second argument some basis $B \subseteq S$ (not necessarily a basis of $S$), which has no influence on the postcondition. However, it carries the essential information responsible for the efficiency of the algorithm.

A new basic operation is required[15]

- *Pivot step* '$\textsf{pivot}(B, p)$': Given a basis $B$ and a point $p \notin \textsf{minB}B$, it returns a basis $B' \subseteq B \cup \{p\}$ with $\rho B' > \rho B$.

Correctness of the procedure is obvious, provided it terminates. The fact that the procedure always terminates follows from the fact, that in the first recursive call the size of the first argument decreases by 1, while in the second call we know

[15] It allows an $O(d^3)$ implementation, although that needs some thinking, cf. [8, 7].

$B, S \subseteq \mathbf{R}^d$.
PRECONDITION:
$B \subseteq S$; $B$ basis.
POSTCONDITION:
$C$ basis of $S$.

```
function Implicit(S, B)
if #S ≤ 2δ then
    C ← basis^d_{2δ} S;
else
    p ←_random S \ B;
    C ← Implicit(S \ {p}, B);
    if p ∉ (C) then
        C ← Implicit(S, pivot(C, p));
return C;
```

that $\rho\mathsf{pivot}(C,p) > \rho B$, and there are only a finite number of radii attained by bases.

As we will see, the efficiency of Implicit stems from the fact that the second argument, the basis $B$, *implicitly enforces* some of the extreme points by excluding them as possible random choices for $p$ in this *and* all recursive calls entailed by the call with $(S, B)$. In fact, we will see that in the second call (if it happens at all) the number of 'available' (i.e. not enforced) extreme points roughly halves on the average. Here are the crucial notions.

**Definition 6 (enforced, hidden dimension).** *$B \subseteq P \subseteq \mathbf{R}^d$, and $B$ basis.*

**(1)** *A point $p \in P$ is called* enforced in $(P, B)$, *if $p$ is contained in all bases $C \subseteq P$ with $\rho C \geq \rho B$.*

**(2)** *The* hidden dimension $\mathrm{hdim}(P, B)$ *of $(P, B)$ is $\delta - i$ where $i$ is the number of enforced elements in $(P, B)$.*

In particular, if $p$ is enforced in $(P, B)$, then $p \in B$.

Next we list a number of simple facts. The reader should confirm their obviousness for her- or himself, so that we have them handy in the proof of the recursion for the number of basis computations.

**Lemma 6.** *$B \subseteq P \subseteq \mathbf{R}^d$, and $B$ basis.*

**(1)** *A point $p \in P$ is enforced in $(P, B)$ iff $\rho(P \setminus \{p\}) < \rho B$.*

**(2)** *Every enforced element in $(P, B)$ is extreme in $P$. There are at most $\delta$ enforced elements in $(P, B)$ and* $\mathrm{hdim}(P, B) \geq 0$.

**(3)** *If* $\mathrm{hdim}(P, B) = 0$, *then $B$ is a basis of $P$.*

**(4)** *If $B \subseteq P' \subseteq P$, then every enforced element in $(P, B)$ is also enforced in $(P', B)$ and* $\mathrm{hdim}(P', B) \leq \mathrm{hdim}(P, B)$.

**(5)** *If $B' \subseteq P$ is a basis with $\rho B' \geq \rho B$ then every enforced element in $(P, B)$ is also enforced in $(P, B')$ and* $\mathrm{hdim}(P, B') \leq \mathrm{hdim}(P, B)$.

*Proof.* (1) ($\Rightarrow$) If $\rho(P \setminus \{p\}) \geq \rho B$, then any basis $C$ of $P \setminus \{p\}$ is a basis with $\rho C \geq \rho B$ and $p \notin C$, and so $p$ is not enforced.

($\Leftarrow$) If $\rho(P \setminus \{p\}) < \rho B$ and $C$ is any basis that does not include $p$ then $\rho C \leq \rho(P \setminus \{p\})$. Hence, $\rho C < \rho B$. Therefore, $\rho C \geq \rho B$ implies $p \in C$.

(2-5) are immediate consequences of (1). ◻

For $k, n \in \mathbf{N}_0$, let $t_k(n)$, and $b_k(n)$ denote the maximum expected number of inclusion tests, and $\mathsf{basis}^d_{2\delta}$-computations, respectively, for a call $\mathsf{Implicit}(S, B)$ with $\#S = n$ and $\mathrm{hdim}(S, B) \leq k$. The number of pivot steps is always one less than the number of basis computations. This follows easily from a proof by induction over recursive calls.

We claim the following recursions (to be proven below).

$$\text{Implicit-recursion, inclusion tests}$$
$$t_k(n) \begin{cases} = 0, & \text{if } n \leq 2\delta, \\ \leq t_k(n-1) + 1 + \frac{1}{n-\delta} \cdot \sum_{i=0}^{k-1} t_i(n), & \text{otherwise.} \end{cases}$$

In particular, $t_0(n) = n - 2\delta$. A proof by induction verifies $t_k(m + 2\delta) \leq 2^k m$ for $k, m \in \mathbf{N}_0$. However, we will see that for $m$ small, a better bound in terms of $k$ can be derived.

$$\text{Implicit-recursion, } \mathsf{basis}^d_{2\delta}\text{-computations}$$
$$b_k(n) \begin{cases} = 1, & \text{if } n \leq 2\delta, \\ \leq b_k(n-1) + \frac{1}{n-\delta} \cdot \sum_{i=0}^{k-1} b_i(n), & \text{otherwise.} \end{cases} \tag{3}$$

In particular, $b_0(n) = 1$.

*Proof of recursion (3) (number of basis computations).* Clearly, $b_k(n) = 1$ for $n \leq 2\delta$. Let $B \subseteq S \subseteq \mathbf{R}^d$, $\#S = n > 2\delta$, $B$ a basis, $\mathrm{hdim}(S, B) \leq k \in \mathbf{N}_0$.

(i) The first recursive call solves a problem of size $n - 1$ and hidden dimension at most $k$.

(ii) There are at most $k$ elements in $S \setminus B$ which trigger a second recursive call: They all have to be extreme elements; there are at most $\delta$ extreme elements, and at least $\delta - k$ of them are enforced in $(S, B)$ (and thus included in $B$).

Let us denote these elements by $p_1, p_2, \ldots, p_r$, $r \leq k$, enumerated in a way that
$$\rho(S \setminus \{p_1\}) \leq \rho(S \setminus \{p_2\}) \leq \cdots \leq \rho(S \setminus \{p_r\}) \ .$$
If $p = p_\ell$ is chosen for the first recursive call (which happens with probability $\frac{1}{\#(S \setminus B)} \leq \frac{1}{n-\delta}$), then we receive $C$, a basis of $S \setminus \{p_\ell\}$. But then,
$$\rho(S \setminus \{p_1\}) \leq \rho(S \setminus \{p_2\}) \leq \cdots \leq \rho(S \setminus \{p_\ell\}) = \rho C < \rho\mathsf{pivot}(C, p) \ .$$
It follows that $p_1, p_2, \ldots, p_\ell$ are enforced in $(S, \mathsf{pivot}(C, p))$, in addition to the previously enforced elements. That is, the hidden dimension of $(S, \mathsf{pivot}(C, p))$ is at most $k - \ell$.

Summing up, the expected number of basis computations entailed by the second recursive call is bounded by
$$\frac{1}{n-\delta} \cdot \sum_{\ell=1}^{r} b_{k-\ell}(n) \leq \frac{1}{n-\delta} \cdot \sum_{i=0}^{k-1} b_i(n) \ .$$

$\square$

For the analysis of $b_k(n)$ we employ a function $\mathbf{N}_0^2 \to \mathbf{R}$ defined by

$$C_k(m) = \begin{cases} 1, & \text{if } k = 0, \\ 0, & \text{if } k > 0 \text{ and } m = 0, \\ C_k(m-1) + \frac{1}{m} \cdot \sum_{i=0}^{k-1} C_i(m-1), & \text{otherwise.} \end{cases}$$

This function will allow us an estimate for the expected number of basis computations via the following two steps.

**Lemma 7.** *$k, n \in \mathbf{N}_0$; $n \geq 2\delta$.*

$$b_k(n) \leq C_k(n - 2\delta + k).$$

**Lemma 8.** *$k, m \in \mathbf{N}_0$.*

$$C_k(m) = \frac{1}{m!} \sum_{i=0}^{m} \binom{k-1}{i-1} \begin{bmatrix} m+1 \\ i+1 \end{bmatrix} \leq \mathrm{e}^{2\sqrt{kH_m}} \ .$$

Here we adopt the convention that, for $j \in \mathbf{N}$, $\binom{j-1}{-1} = 0$ and $\binom{-1}{-1} = 1$.

*Proof by induction of Lemma 7.* For $k = 0$,

$$b_0(n) = 1 \leq C_0(n - 2\delta)$$

holds as required, since $C_0(n - 2\delta) = 1$.

For $k \geq 1$ and $n = 2\delta$, the assertion claims

$$b_k(2\delta) = 1 \leq C_k(k) \ ,$$

which holds since $C_k(k) \geq C_k(k-1) \geq \cdots \geq C_k(1) = C_k(0) + C_0(0) = 1$.

For $n > 2\delta, k > 0$

$$\begin{aligned} b_k(n) &\leq b_k(n-1) + \frac{1}{n-\delta} \sum_{i=0}^{k-1} b_i(n) \\ &\leq C_k(n - 1 - 2\delta + k) + \frac{1}{n-\delta} \sum_{i=0}^{k-1} C_i(n - 2\delta + i) \\ &\leq C_k(n - 2\delta + k - 1) + \frac{1}{n - 2\delta + k} \sum_{i=0}^{k-1} C_i(n - 2\delta + k - 1) \\ &= C_k(n - 2\delta + k) \ . \end{aligned}$$

We have used induction, the inequality $\frac{1}{n-\delta} \leq \frac{1}{n-2\delta+k}$ (since $k \leq \delta$), and monotonicity of $C_i(\cdot)$: $C_i(\ell) \leq C_i(\ell+1)$. $\square$

*Combinatorial Interpretation of $C_k(m)$.* It turns out that $C_k(m)$ has a combinatorial interpretation, leading to the closed form in Lemma 8. For $n \in \mathbf{N}$, $k \in \mathbf{N}_0$, we consider the set of permutations $\mathcal{S}_n$, and we define the set $\mathcal{S}_n^{(k)}$ of *$k$-decorated* permutations. A $k$-decorated permutation is a pair $(\pi, (j_1, j_2, \ldots, j_{h-1}))$, where $\pi \in \mathcal{S}_n$, $h$ is the number of cycles of $\pi$, and $j_1, j_2, \ldots, j_{h-1}$ are positive integers summing up to $k$.

Assume some canonical ordering[16] of the cycles of a permutation in $\mathcal{S}_n$, where the last cycle is the one containing $n$. Then we can view a $k$-decorated permutation in $\mathcal{S}_n^{(k)}$ as a permutation whose $i$th cycle is labeled by a positive integer $j_i$, except for the last one, which gets 0, and the sum of all the labels is $k$. Here comes the combinatorial interpretation.

**Lemma 9.** $k, m \in \mathbf{N}_0$.

$$m!\, C_k(m) = \#\mathcal{S}_{m+1}^{(k)} \ .$$

*Proof.* We argue that $c_k(m) := \#\mathcal{S}_{m+1}^{(k)}$ satisfies the same recurrence relation as $m!\, C_k(m)$.

$$c_0(m) = m! = m!\, C_0(m) \ , \quad \text{for } m \in \mathbf{N}_0,$$

because we can 0-decorate a permutation in $\mathcal{S}_{m+1}$ only if there is exactly one cycle; there are $m!$ such permutations in $\mathcal{S}_{m+1}$.

$$c_k(0) = 0 = 0!\, C_k(0) \ , \quad \text{for } k \in \mathbf{N},$$

because the unique permutation in $\mathcal{S}_1$ has one cycle, and therefore cannot be $k$-decorated with positive $k$.

For $m, k \in \mathbf{N}$, we split $\mathcal{S}_{m+1}^{(k)}$ into two sets. The first set contains the $k$-decorated permutations with $m+1$ appearing in a cycle of its own. For each $i$-decorated permutation in $\mathcal{S}_m^{(i)}$, $i = 0, 1, \ldots, k-1$, there is exactly one corresponding $k$-decorated permutation in $\mathcal{S}_{m+1}$ with $m+1$ appearing in a cycle (labeled 0) of its own, and the cycle containing $m$ labeled $k-i$. Consequently, the first set contains

$$\sum_{i=0}^{k-1} c_i(m-1) \tag{4}$$

$k$-decorated permutations.

The second set contains the $k$-decorated permutations where $m+1$ appears in some nontrivial cycle. There are $m$ ways to integrate $m+1$ into existing cycles of some permutation in $\mathcal{S}_m^{(k)}$. We let the cycles containing $m$ and $m+1$, resp. switch their labels (unless $m$ and $m+1$ appear in the same cycle). Hence, this second set has size

$$m \cdot c_k(m-1) \ . \tag{5}$$

Combining (4) and (5) we obtain

$$c_k(m) = m \cdot c_k(m-1) + \sum_{i=0}^{k-1} c_i(m-1) \ ,$$

[16] For example, order the cycles in increasing order according to their largest element.

the recurrence relation for $m!\,C_k(m)$. ◻

$i \in \mathbf{N}_0$. A permutation with $i+1$ cycles gives rise to exactly $\binom{k-1}{i-1}$ $k$-decorated permutations, since $k$ can be written in that number of ways as an ordered sum of $i$ positive integers[17]. Thus,

$$C_k(m) = \frac{1}{m!}\sum_{i=0}^{m}\binom{k-1}{i-1}\begin{bmatrix} m+1 \\ i+1 \end{bmatrix}, \qquad \text{for } k, m \in \mathbf{N}_0,$$

and the identity in Lemma 8 is proven.

*Upper Bound for $C_k(m)$.* For the upper estimate we start using Lemma 10 in the appendix.

$$\begin{aligned} C_k(m) &= \frac{1}{m!}\sum_{i=0}^{k}\binom{k-1}{i-1}\begin{bmatrix} m+1 \\ i+1 \end{bmatrix} \\ &\le \frac{1}{m!}\sum_{i=0}^{k}\binom{k-1}{i-1}\frac{m!}{i!}(H_m)^i \\ &= \sum_{i=0}^{k}\binom{k-1}{i-1}\frac{(H_m)^i}{i!} \;\le\; \sum_{i=0}^{k}\binom{k}{i}\frac{(H_m)^i}{i!} \\ &\le \sum_{i=0}^{k}\frac{(kH_m)^i}{(i!)^2} \qquad \text{since } \tbinom{k}{i} \le k^i/i! \\ &= \sum_{i=0}^{k}\left(\frac{(\sqrt{kH_m})^i}{i!}\right)^2 \;\le\; \left(\sum_{i=0}^{\infty}\frac{(\sqrt{kH_m})^i}{i!}\right)^2 \\ &= \mathsf{e}^{2\sqrt{kH_m}}\,. \end{aligned}$$

This establishes the upper bound claimed in Lemma 8.

**Theorem 4.** *$n \in \mathbf{N}$; $P \subseteq \mathbf{R}^d$, $\#P = n \ge 2(d+1)$.*
*A call* $\mathsf{Implicit}(P, \emptyset)$ *computes a basis of $P$ with an expected number of at most*

$$\mathsf{e}^{2\sqrt{(d+1)H_{n-d-1}}} \qquad \mathsf{basis}^d_{2(d+1)}\text{-}\textit{computations (pivot steps, resp.)}$$

*and an expected number of at most*

$$(n-2(d+1))\,\min\{\mathsf{e}^{2\sqrt{(d+1)H_{n-d-1}}}, 2^{d+1}\} \qquad \textit{inclusion tests.}$$

The bound for inclusion tests follows directly from the fact that no basis is involved in more than $n-2(d+1)$ inclusion tests, and the simple bound concluded right after setting up the recursion for $t_k(n)$.

[17] Note that, indeed, 0 can be written in $1 = \binom{-1}{-1}$ ways as the sum of 0 positive integers.

## 5 Bibliographical Remarks and Discussion

We have already indicated (and the reader might have suspected) that the methods described in this tutorial apply to problems beyond smallest enclosing balls. In fact, there is a host of other problems for which we have a 'Lemma 1', the most prominent one being *linear programming.*

In the general framework of *LP-type problems* (which is actually motivated by the linear programming case), we rank subsets $S$ of some ground set $P$ by an objective function[18] $w : 2^P \to \mathcal{O}$, $\mathcal{O}$ an ordered set, and we require $(P, w)$ to satisfy the conditions of Lemma 1 (2)–(4).

**Definition 7 (LP-type problem).** *$P$ finite set; $w : 2^P \mapsto \mathcal{O}$, $\mathcal{O}$ ordered set; $\delta \in \mathbf{N}_0$.*

*$(P, w)$ is an* LP-type problem of combinatorial dimension at most $\delta$ *if the following three axioms are satisfied for all $S \subseteq P$.*

**(1)** *If $S' \subseteq S$ then $wS' \leq wS$.*

**(2)** *Let $S' \subseteq S$. Then $wS' = wS$ iff $w(S' \cup \{p\}) = wS'$ for all $p \in S \setminus S'$.*

**(3)** *If $wS' \neq wS$ for all proper subsets $S'$ of $S$, then $\#S \leq \delta$.*

There are other abstractions of linear programming along these lines. Kalai has introduced so-called *abstract objective functions*, which are special LP-type problems [12]. There is also the framework of *abstract optimization problems* which is still more general than that of LP-type problems [8].

By Lemma 1, the smallest enclosing ball problem gives rise to an LP-type problem. In linear programming, $P$ is a set of inequality constraints (halfspaces in $\mathbf{R}^d$), and for $S \subseteq P$, $wS$ is defined as the lexicographically smallest point in the intersection of all halfspaces in $S$, where $w$-values are to be compared lexicographically [14].

Another interesting (nonlinear) LP-type problem is obtained from a set $P$ of points in $\mathbf{R}^d$, by letting $wS := -\inf_{x \in \mathsf{conv} S} \|x\|$ (i.e. the negative distance between the origin and $\mathsf{conv}S$), [14, 8]. Many more geometric optimization problems fit into the framework [9].

The significance of the abstract framework stems from the fact that the algorithm $\mathsf{Implicit}$ we have described above for smallest enclosing balls actually works (with the same analysis) for arbitrary LP-type problems, provided the problem-specific primitives are supplied:

- *Violation test* '$w(C \cup \{p\}) > w(C)$'.
- *Pivot step* '$\mathsf{pivot}(B, p)$': Given a basis $B$ and an element $p$ with $w(B \cup \{p\}) > wB$, it returns a basis $B' \subseteq B \cup \{p\}$ with $wB' > wB$.

In addition, the call to $\mathsf{basis}^d_{2\delta}\, S$ needs to be replaced by some method for finding a basis of a set with at most $2\delta$ elements. Even if this is done by brute force, $\mathsf{Implicit}$ is an expected $O(n)$ algorithm for solving an LP-type problem of constant combinatorial dimension over an $n$-element ground set. This is remarkable,

[18] $wS = \rho S$ in case of smallest enclosing balls.

because even for concrete LP-type problems like linear programming, linear-time algorithms have only been found in the eighties. Most notably, Megiddo published the first (deterministic) $O(n)$ algorithm for linear programming in fixed dimension in 1984 [15], with a $2^{2^d}$ dependence on the dimension $d$. A generalization of the $O(n)$ bound to certain convex programming problems, including the smallest enclosing ball problem, is due to Dyer [4].

By using randomization, simple $O(n)$ algorithms for linear programming have then been found by Clarkson [2] and Seidel [17]. Subsequent developments mainly consisted of attempts to reduce the dependence of the runtime on the dimension $d$. For linear programming, the first subexponential bounds have independently been found by Kalai [13] as well as Matoušek, Sharir and Welzl [14]. The notion of $k$-decorated permutations we have used in the analysis of the subexponential algorithm $\mathsf{Implicit}$ is also due to Kalai [12].

Combining these algorithms with the ones by Clarkson, one obtains the fastest known combinatorial algorithm for linear programming, of runtime (i.e. number of arithmetic operations)

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

see [9]. The currently best deterministic bound is $O(d^{O(d)}n)$ [1].

As far as other LP-type problems (in particular the smallest enclosing ball problem) are concerned, the situation is similar, although there are additional technical obstacles.

Exactly these obstacles made us formulate the procedure $\mathsf{Implicit}(S, B)$ in such a way that the recursion bottoms out when $\#S \leq 2\delta$. The reader might have noticed that as long as there are elements in $S \setminus B$, we could continue with the recursive call $\mathsf{Implicit}(S \setminus \{p\}, B)$, for $p$ randomly chosen in $S \setminus B$. In fact, we can, but then a subexponential bound for the expected number of basis computations does no longer hold. For linear programming, it still works out: because *every* basis has size $d$ in the above formulation of the problem, the recursion bottoms out when $\#S = d$, in which case the basis computation is trivial; this 'saves' the subexponential bound, and the algorithm described in [14] is actually this variant of the algorithm $\mathsf{Implicit}$.

In the general case, a problem of size $\delta$ is not trivial. On the contrary, in case of smallest enclosing balls, we have made quite some effort to solve a problem involving $\delta = d + 1$ points with an expected number of $O(\left(\frac{3}{2}\right)^d)$ steps.

The subexponential bound can be saved in general, though, because there is an algorithm to solve an LP-type problem over $O(\delta)$ elements with an expected number of $\mathsf{e}^{O(\sqrt{\delta}\,)}$ primitive operations by Gärtner [8]. This algorithm is more complicated than the ones discussed here, but it lets us obtain essentially the same bounds for arbitrary LP-type problems as we have them for linear programming.

The procedures $\mathsf{Explicit}$ as well as $\mathsf{ForceOrNot}$ are interesting in the sense that they do *not* work for general LP-type problems. $\mathsf{Explicit}$ has first been described in [19], and it is actually modeled after Seidel's linear programming algorithm [17]. In order for these algorithms to work, we need a notion of explicitly 'forcing' elements on the 'boundary'. Such a notion does not exist for general LP-type problems. However, the framework can be specialized to include problems for which this forcing makes sense, in which case the algorithms (that can then be interpreted as primal-dual methods) apply [10].

## References


1. Bernard Chazelle, Jiří Matoušek: On linear-time deterministic algorithms for optimization problems in fixed dimensions, J. Algorithms **21** (1996), 579–597.
2. Kenneth L. Clarkson: A Las Vegas algorithm for linear and integer programming when the dimension is small, J. Assoc. Comput. Mach. **42**(2) (1995), 488–499.
3. Martin E. Dyer: Linear algorithms for two and three-variable linear programs, SIAM J. Comput. **13** (1984), 31–45.
4. Martin E. Dyer: On a multidimensional search technique and its application to the Euclidean one-center problem, SIAM J. Comput. **15** (1986), 725–738.
5. Jürgen Eckhoff: Helly, Radon, and Carathéodory type theorems, in *Handbook of Convex Geometry* (P.M.Gruber, J.M.Wills, eds.), Vol. **A** (1993), 389-448, North Holland.
6. Bernd Gärtner, Michael Hoffmann, Sven Schönherr: Geometric optimization, *Reference Manual of the Computational Geometry Algorithms Library (CGAL)*, Release 2.2 (`www.cgal.org`) (2000).
7. Bernd Gärtner, Sven Schönherr: An efficient, exact, and generic quadratic programming solver for geometric optimization, Proc. 16th Ann. ACM Symp. Computational Geometry (2000), 110–118.
8. Bernd Gärtner: A subexponential algorithm for abstract optimization problems, SIAM J. Comput. **24** (1995), 1018–1035.
9. Bernd Gärtner, Emo Welzl: Linear Programming – Randomization and abstract frameworks, Proc. 13th Ann. ACM Symp. Theoretical Aspects of Computer Science (1996), 669–687.
10. Bernd Gärtner, Emo Welzl: LP-type problems of the second kind for primal-dual methods, manuscript, in preparation (2001).
11. Ronald L. Graham, Donald E. Knuth, Oren Patashnik: *Concrete Mathematics; A Foundation for Computer Science*, Addison-Wesley (1989).
12. Gil Kalai: Linear programming, the simplex algorithm and simple polytopes, Math. Programming **79** (1997), 217–233.
13. Gil Kalai: A subexponential randomized simplex algorithm, Proc. 24th Ann. ACM Symp. Theory of Computing (1992), 475–482.
14. Jiří Matoušek, Micha Sharir, Emo Welzl: A subexponential bound for linear programming, Algorithmica **16** (1996), 498–516.
15. Nimrod Megiddo: Linear programming in linear time when the dimension is fixed, J. Assoc. Comput. Mach. **31** (1984), 114–127.
16. Rajeev Motwani, Prabhakar Raghavan: *Randomized Algorithms*, Cambridge University Press (1995).
17. Raimund Seidel: Small-dimensional linear programming and convex hulls made easy, Discrete Comput. Geom. **6** (1991), 423–434.
18. James Joseph Sylvester: A question in the geometry of situation, Quart. J. Math. **1** (1857), 79.
19. Emo 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.

## A An Estimate for Cycle Numbers

**Lemma 10.** *$i, n \in \mathbf{N}_0$; $i \leq n$.*

$$\begin{bmatrix} n+1 \\ i+1 \end{bmatrix} \leq \frac{n!}{i!}(H_n)^i \ .$$

*Proof by induction.*
$i = 0$:

$$\begin{bmatrix} n+1 \\ 1 \end{bmatrix} = n! = \frac{n!}{0!}(H_n)^0 \ .$$

$i = n$:

$$\begin{bmatrix} n+1 \\ n+1 \end{bmatrix} = 1 \leq \frac{n!}{n!}(H_n)^n \ .$$

$0 < i < n$:

$$\begin{aligned} \begin{bmatrix} n+1 \\ i+1 \end{bmatrix} &= n \begin{bmatrix} n \\ i+1 \end{bmatrix} + \begin{bmatrix} n \\ i \end{bmatrix} \\ &\leq n \frac{(n-1)!}{i!}(H_{n-1})^i + \frac{(n-1)!}{(i-1)!}(H_{n-1})^{i-1} \\ &= \frac{n!}{i!}\left((H_{n-1})^i + i\,\frac{1}{n}(H_{n-1})^{i-1}\right) \\ &\leq \frac{n!}{i!}\left(H_{n-1} + \frac{1}{n}\right)^i = \frac{n!}{i!}\,(H_n)^i \ , \end{aligned}$$

where we employed the Binomial Theorem. $\square$