Geometric Algorithms and Algorithmic Geometry

László Lovász

Department of Computer Science, Eötvös University, H-1088 Budapest, Hungary and
Department of Computer Science, Princeton University, Princeton, NJ 08544, USA

0. Introduction

In this paper I would like to illustrate two facts. First, that ideas borrowed from convex, projective, and other classical branches of geometry play an important role in the design of algorithms for problems that do not seem to have anything to do with geometry: for problems in optimization, combinatorics, algebra, and number theory. Second, that such applications of geometry suggest some very elementary algorithmic questions concerning geometric notions, whose solution is far from complete. How to compute the volume? How to decide whether or not a convex body contains the other? How to present a convex body as an input to an algorithm? Even partial answers to these problems require a wealth of mathematical ideas, often (again) quite unrelated to the original question. On the other hand, answers to these questions have a very wide range of applicability.

In the first chapter some typical constructions are surveyed which lead from non-geometric problems to geometric ones. Such is, of course, the classical field of the “Geometry of Numbers”. This is the application of lattice geometry and convexity to number theory, and was initiated by Minkowski around the turn of the century. We only touch upon this area to point out that the recent shift of interest from structural problems to algorithmic ones has induced a lot of activity, and this new approach has even fertilized classical types of investigations.

We also give a brief introduction to polyhedral combinatorics, developed in the 60s by Ford, Fulkerson, Hoffman, Edmonds, and others. Here polyhedral theory and linear programming are applied to combinatorial optimization problems. This approach yields surprisingly successful algorithms both from a theoretical and practical point of view. We also show how some enumeration problems are related to the computation of the volume of certain polytopes.

The second chapter treats various forms of presentation of a convex body to an algorithm. This seemingly technical issue leads to a powerful equivalence principle between different ways of presentation. A combination of this principle with polyhedral combinatorics provides (at least theoretically) efficient algorithms for most combinatorial optimization problems that can be solved efficiently by any other means.

Chapter 3 describes a basic construction in algorithmic geometry, namely the Löwner-John ellipsoids. These are used in the ellipsoid method of Shor, Yudin, Nemirovskii and Khachiyan (in particular in establishing the equivalence principle formulated in the previous chapter) and in many other geometric
algorithms. We also sketch the geometric background of Karmarkar's celebrated linear programming algorithm.

In the last chapter we discuss recent developments concerning the problem of computing the volume of a convex body. After some very discouraging negative results, Dyer, Frieze and Kannan designed a polynomial time randomized algorithm which approximates the volume with an arbitrarily small error. It turns out that the crucial issue is to generate a random point uniformly distributed over a general convex body. The solution of this problem leads to Markov chains, eigenvalues of matrices, differential geometry, and even to some algebraic topology.

There is an extremely important branch of algorithmic geometry which is not treated here; this is usually called computational geometry. Polyhedral combinatorics leads to high-dimensional problems; other — more immediate — applications in image processing and robotics lead to two- and three-dimensional questions. In such cases, a different notion of efficiency is the crucial one. Our concern will be polynomial time; in computational geometry, usually linear or almost-linear time is the target. The interested reader may consult the monographs by Preparata and Shamos (1985) and by Edelsbrunner (1987).

1. Number Theory, Combinatorics, and Convex Sets

In this chapter we illustrate some of the most important constructions which transform algorithmic problems in various branches of mathematics into geometric questions, and in fact into very simple ones. These examples will also serve to help us put the corresponding general algorithmic problems into the right framework.

1.1 Geometry of Numbers

This is the classical area where the application of geometric ideas to non-geometric problems has been very successful ever since Minkowski's work. In the last decade, algorithmic questions arising from numerical methods, primality testing, computational algebra, cryptography, and other areas have revigorated the field; it turns out that to solve these algorithmic problems, often new structural insight is needed.

The classical problem in number theory which lead Minkowski to the "Geometry of Numbers" is the problem of simultaneous diophantine approximation: given \( n \) real numbers \( \alpha_1, \ldots, \alpha_n \), and an "error bound" \( \varepsilon > 0 \), find integers \( p_1, \ldots, p_n \) and \( q \) such that \( q > 0 \) and

\[
\left| \alpha_i - \frac{p_i}{q} \right| \leq \frac{\varepsilon}{q} \quad (i = 1, \ldots, n).
\]

The answer is trivial for \( \varepsilon = 1/2 \) (we can choose \( q \) arbitrarily); it was proved by Dirichlet that such integers exist for every \( \varepsilon > 0 \) and in fact we can require that \( q \leq \varepsilon^{-n} \).

While the proof of Dirichlet's theorem is quite easy (using "Dirichlet's Principle"), no efficient algorithm is known to find such an approximation. A geometric
translation is useful, among others, in finding a solution with a worse bound on $q$.

Consider the following vectors in $\mathbb{R}^{n+1}$: $e_1, \ldots, e_n$ (the first $n$ basis vectors) and $(\alpha_1, \ldots, \alpha_n, 1)^T$. The linear combinations of these vectors with integer coefficients form a lattice $L$, whose typical point looks like $(q\alpha_1 - p_1, \ldots, q\alpha_n - p_n, q)^T$ with integral $p_1, \ldots, p_n$ and $q$. Dirichlet's Theorem is equivalent to asserting that the lattice $L$ has a non-zero lattice point in the brick

$$K = [-e, e] \times \ldots \times [-e, e] \times [-e^{-n}, e^{-n}].$$

Minkowski's famous "First Theorem" shows that the fact that $K$ is a brick is irrelevant: all that matters is that $K$ is a convex body centrally symmetric with respect to the origin with volume at least $2^{n+1}$ times the determinant of the lattice.

The convex body $K$ can be viewed as the unit ball of a norm, and so Minkowski's theorem guarantees the existence of a "short" non-zero lattice vector in every lattice, measured in an arbitrary norm.

Many other problems turn out equivalent to the existence of a short non-zero lattice vector in appropriately defined lattices. For example, let $f(x) = \sum_{i=1}^n a_i x^i$ be a polynomial with integral coefficients; we want to know whether $f$ is irreducible. Let $\alpha$ be a root of $f$; for simplicity, assume that $\alpha$ is real. Let $K$ be a sufficiently large real number (computable from $f$) and let $L \subseteq \mathbb{R}^{n+1}$ be the lattice generated by the vectors $e_i + K\alpha^{i-1} e_{n+1}$, $1 \leq i \leq n$. If $f$ is reducible over the rational field, then $L$ contains a non-zero vector with (euclidean) length at most $2^n \sqrt{\sum_i \alpha_i^2}$. On the other hand, if $f$ is irreducible then every non-zero vector in the lattice is at least $2^n$ times this length. This fact is the basis of the efficient (polynomial time) algorithm for factoring polynomials (Lenstra, Lenstra and Lovász 1982).

We shall not go into geometric algorithms involving lattices in this survey; instead, we refer to (Lovász 1989) for a survey.

### 1.2 Polyhedral Combinatorics

Polyhedral combinatorics provides perhaps the most successful general approach to various combinatorial problems. To illustrate the idea, consider the following simple graph-theoretic problem. Let $G$ be a (finite) graph with node set $V$ and edge set $E$. A set of nodes of $G$ is called *stable* if no two elements of it are connected by an edge. Let $\alpha(G)$ denote the maximum cardinality of a stable set in $G$. To determine $\alpha(G)$ is difficult (NP-hard) in general; but the approach of polyhedral combinatorics suggests efficiently solvable special cases, as well as efficiently computable estimates of $\alpha(G)$.

Let us construct the following convex polytope: for every stable set $S$ of nodes, let $\chi^S$ denote the incidence vector of $S$ (in the space $\mathbb{R}^V$ of vectors indexed by the nodes of $G$), and let $\text{STAB}(G)$ be the convex hull of such incidence vectors. Then

$$\alpha(G) = \max\{|S| : S \text{ stable}\} = \max\{1 \cdot \chi^S : S \text{ stable}\} = \max\{\sum_i x_i : x \in \text{STAB}(G)\}.$$

(since the maximum of a linear objective function over a polytope is automatically assumed at a vertex).

The hope is to apply the powerful methods of linear programming to find this maximum. To this end, however, we have to find a representation of $\text{STAB}(G)$ as
the solution set of a system of linear inequalities. Such a representation exists of course; but how to find it? Note that the arguments presented so far would work for the problem of finding a maximum cardinality member in any collection of subsets of a finite set \( V \); where we have to be problem-specific is in finding the linear representation of the corresponding polytope.

There is a very large number of results presenting linear descriptions of various combinatorial polyhedra; we shall restrict ourselves to the stable set polytope and be content with giving a couple of illustrations. A natural starting point is the following set of linear inequalities:

\[
\begin{align*}
0 \leq x_i & \leq 1 \quad (i \in V), \\
x_i + x_j & \leq 1 \quad (ij \in E).
\end{align*}
\]

It is clear that the incidence vector of any stable set, and therefore every vector in \( STAB(G) \), satisfies inequalities (1) and (2). The solution set of this system is, however, a larger polytope than \( STAB(G) \) in general. In fact, inequalities (1) and (2) suffice to describe \( STAB(G) \) if and only if \( G \) is a bipartite graph. So at least for bipartite graphs, \( \alpha(G) \) can be determined using linear programming algorithms. Or, applying the Duality Theorem of linear programming, one can obtain a min-max formula for \( \alpha(G) \) (which, after some transformations, turns out to be equivalent to the König Theorem).

If the graph is non-bipartite then it contains a circuit of odd length and we can use such a circuit to add further constraints to (1) and (2): for every odd circuit \( C \) we write up the inequality

\[
\sum_{i \in V(C)} x_i \leq \frac{|C| - 1}{2}.
\]

Graphs for which (1), (2) and (3) suffice to describe \( STAB(G) \) are called \( t \)-perfect; these graphs are less well understood than bipartite graphs, but several important classes of them are known. We mention series-parallel graphs, i.e., graphs which can be obtained by the repeated application of series and parallel extensions (Chvátal 1975, Boulala and Uhry 1979).

Assume that \( G \) is \( t \)-perfect; then \( \alpha(G) \) can be expressed as the optimum value of a linear program with constraints (1)–(2)–(3). Note, however, that (3) includes possibly exponentially many constraints (in \( n = |V| \)), and so writing up this program and calling a linear program solver would be inefficient. We shall see that general geometric considerations provide efficient solution methods for such systems which do not need the whole system explicitly.

Further natural inequalities valid for \( STAB(G) \), but not implied by the previous ones, can be written up. Let \( B \) be a clique (a set of mutually adjacent nodes in \( G \)). Then every stable set meets \( B \) in at most one node and hence the inequality

\[
\sum_{i \in B} x_i \leq 1
\]

is valid for \( STAB(G) \). Those graphs for which (1) and (4) suffice to describe \( STAB(G) \) are called perfect. This rich class has been defined by Berge (1961) before these polyhedral methods were introduced, motivated by many classes of examples. Let us mention one: Let \((V, \leq)\) be a (finite) partially ordered set.
Define a graph $G$ by connecting two elements of $V$ iff they are comparable. \textit{Comparability graphs} obtained this way are perfect.

It is beyond the scope of this paper to treat perfect graphs; we refer to Berge and Chvátal (1984) and Grötschel, Lovász and Schrijver (1988). It should be mentioned, however, that polyhedral methods play a central role in their study, even in proving seemingly elementary properties.

Polyhedral combinatorics is closely related to \textit{integer linear programming}. The basic problem here is to solve a system of linear inequalities \textit{in integers}. For example, the integer solutions of the system (1)–(2) are exactly the incidence vectors of stable sets. The problem of simultaneous diophantine approximation can also be viewed as an integer programming problem: Given $\alpha_i, e$, and $Q$, find a solution of

$$-e \leq q \alpha_i - p_i \leq e \quad (i = 1, \ldots, n),$$

$$1 \leq q \leq Q.$$ 

For a long while, integer programming and lattice geometry have developed independently, and used rather different methods. A first substantial connection was established by Lenstra (1983), who designed a polynomial time algorithm to solve integer linear programs with a bounded number of variables, using methods borrowed from the geometry of numbers. This approach to integer programming seems to gain further momentum in recent years.

\subsection{1.3 Enumeration and Volume}

Some enumeration problems also have useful translations into geometry. This connection is not so well understood and we only give one example. Let $P = (E, \leq)$ be a partially ordered set. A linear order of $E$ which is compatible with the given partial order is called a \textit{linear extension} of the partial order. The number of linear extensions is a measure of how incomplete the partial order is, and it plays an important role in several algorithmic and other questions. There is no reasonable formula or efficient algorithm known to find this number. In fact, no such algorithm can be expected by the recent important result of Brightwell and Winkler (1990), which asserts that to determine the number of linear extensions of a poset is \#P-complete. But an efficient approximation algorithm can be obtained, which is based on the following construction (Stanley 1986).

Consider the linear space $\mathbb{R}^E$ and the incidence vectors of filters in $P$. The convex hull of these incidence vectors is a polytope $\text{FILT}(P)$. By the methods of polyhedral combinatorics mentioned in the previous section, it can be shown that $\text{FILT}(P)$ is defined by the inequalities

$$0 \leq x_i \leq 1 \quad (i \in E),$$

$$x_i \leq x_j \quad (i \leq j).$$

This fact can be used, as sketched above, to solve optimization problems involving filters. Right now, however, the following fact is important: \textit{the number of linear extensions of $P$ equals $n!$ times the volume of $\text{FILT}(P)$}. This observation reduces the problem of enumerating linear extensions to the problem of determining the volume of a polytope, which is described as the solution set of a small number of simple linear inequalities (in $n$-space we have $O(n^2)$ inequalities). We shall return to this general geometric problem in Chapter 4.
2. What is a Convex Body?

We have seen that various number theoretic, algebraic, and combinatorial questions can be reduced to quite fundamental problems in geometry, such as computing the volume or finding the maximum of a linear objective function over a convex body. Before treating geometric algorithms to solve such problems, we have to introduce the right framework.

A convex body is a closed, bounded, full-dimensional convex set in \( \mathbb{R}^n \). This simple definition becomes insufficient, however, if we are interested in algorithmic questions. In this chapter we discuss our aspects for algorithms and the algorithmic notion of a convex body.

2.1 Basic Algorithmic Problems for a Convex Body

Our condition for the "efficiency" of an algorithm is worst-case polynomial time: this means that there exists a constant \( c > 0 \) such that for every input of length \( n \), the algorithm makes \( O(n^c) \) bit-operations. Here the length of the input is the total number of bits needed to describe the input. We assume that input numbers are always rational, and their contribution to the input length is the total number of digits in the binary representation of the numerator and of the denominator.

For the theory of polynomial time algorithms, and for related notions in complexity theory like NP and NP-hard, see e.g. (Garey and Johnson 1979). To follow this paper, it should suffice to translate "NP-hard" or "\#P-complete" as "it is hopeless to find a polynomial-time algorithm for this problem".

We want to study algorithms whose input is a convex body, and with this we run into trouble. How to present a convex body? Various forms occur in various problems. In linear programming, one always considers convex polyhedra, presented as the solution set of a system of linear inequalities. An equally natural form in which convex bodies come up is a convex polytope, presented as the convex hull of an explicitly given set of vectors. In Banach space theory, convex bodies arise as unit balls of norms, where the norm may be given by some formula; e.g. \( \sum_i |x_i|^3 \leq 1 \) defines the unit ball of the \( \ell_3 \)-norm. In geometry, a convex body \( K \subseteq \mathbb{R}^n \) is sometimes described by its support function: this is essentially the function \( q_K : \mathbb{R}^n \to \mathbb{R} \) defined by \( q_K(u) = \max\{u \cdot x : x \in K\} \).

The polytope \( \text{STAB}(G) \) is defined as the convex hull of a set of vectors (incidence vectors of stable sets), but to describe it, it would be very inefficient to list these vectors; it suffices to specify the graph \( G \). So \( \text{STAB}(G) \) is presented by implicitly specifying its vertices.

We are interested in algorithms that are as independent of the specifics of the presentation of the body as possible. It turns out that most geometric algorithms depend on the possibility to carry out one or more of the following tasks:

- **Membership Test**: Given a (rational) vector \( x \), decide whether or not \( x \in K \).
- **Separation**: Given a (rational) vector \( x \), decide whether or not \( x \in K \), and if not, find a hyperplane separating \( x \) from \( K \).
- **Validity Test**: Given a linear inequality \( a \cdot x \leq \alpha \) (with rational coefficients), decide whether or not the inequality is valid for all \( x \in K \).
- **Violation**: Given a linear inequality \( a \cdot x \leq \alpha \) (with rational coefficients), decide whether or not the inequality is valid for all \( x \in K \), and if not, find an \( x \in K \) violating it.
It is clear that finding a separating hyperplane is a more difficult task than testing for membership, and finding a violating point is more difficult than testing for validity. Scanning through the examples above, we see that depending on the presentation of $K$, one or the other of these tasks can be carried out easily while others appear non-trivial. For example, if the body is given as the solution set of linear inequalities, then it is trivial to test membership by substituting $x$ in each of the defining inequalities; and if it violates one, this also yields a separating hyperplane. On the other hand, there is no obvious way to test the validity of an inequality. Similarly, if the body is presented as the convex hull of vectors, then it is trivial to test validity of a linear inequality, but testing membership is non-trivial.

For STAB($G$), neither one of the above tasks is easy, at least if we allow only polynomial time in $n$ (note that the input size in $O(n^2)$). In fact, each of the above tasks is NP-hard.

To get presentation-independent results, we define a membership oracle as a black box which works as follows: if we plug in a vector $x \in \mathbb{R}^n$, it returns "YES" or "NO". Its answers must be consistent with the interpretation that "YES" means $x \in K$ for some convex body $K$. We can call such an oracle in an algorithm; one call counts as a single step. Of course, if we have an algorithm to test membership in polynomial time, then we can "put this inside the black box", and this increases the running time by a polynomial factor only.

One can introduce separation, validity, and violation oracles in a similar way.

The following is a surprising and powerful principle (not a theorem!):

**Equivalence Principle.** The four oracles above are equivalent from the point of view of polynomial time algorithms.

In other words, if for a class of convex bodies we have a polynomial time algorithm to solve either one of them, then all the others can be solved in polynomial time. This principle is true only under some technical assumptions and restrictions which, however, seldom cause any problem in its applications. There are various ways to formulate technical conditions making the Equivalence Principle valid; we only sketch some, and refer for a complete discussion to the monograph (Grötschel, Lovász and Schrijver 1988).

One set of these technicalities consists of making the "boundedness" and "full-dimensionality" properties of convex bodies effective: we need to know a number $R > 0$ such that $K$ is contained in the ball with radius $R$ about the origin, and another number $r > 0$ such that $K$ contains a ball with radius $r$. These numbers must be considered part of the input to any algorithm, so the number of bits needed to write them down must be included in the input size. If the body is given by a membership oracle, and we want to solve the other tasks, then in addition the center of an inscribed ball with radius $r$ must be given in advance. It is easy to argue that without this kind of information, the Equivalence Principle would not be valid.

Another set of limitations comes from numerical errors. We have to re-define the oracles so that an "error bound" $\varepsilon > 0$ is also part of the input, and then allow a small error in the answer. For example, in a membership oracle we should allow either a "YES" or a "NO" answer if the distance of $x$ from the boundary of $K$ is less than $\varepsilon$. One can formulate "weak" versions of all the above oracles in an analogous way. A precise form of the Equivalence Principle holds for these
weak versions. (If the convex body $K$ is a polytope with (say) 0-1 vertices, then we do not have to restrict ourselves to weak versions.)

The main ingredient in the proof of the Equivalence Principle is the Ellipsoid Method, to be sketched in the next chapter.

### 2.2 Applications of the Equivalence Principle

Some consequences of the Equivalence Principle are immediate. Since testing membership in a polyhedron presented by an (explicit) system of linear inequalities is trivial, we can test validity of a linear inequality for such polyhedra in polynomial time. This result implies a polynomial time test for the solvability of a system of linear inequalities, which in turn implies a polynomial time algorithm to solve linear programs.

In a similar way it follows that quadratic programs with positive definite constraints can be solved in polynomial time.

Several basic algorithmic issues concerning convex bodies can be solved using just the Equivalence Principle; let us mention two of these. The issue of validity of a linear constraint can be viewed as a special case of either of the following questions:

- Given two convex bodies $K_1$ and $K_2$, are they disjoint?
- Given two convex bodies $K_1$ and $K_2$, is $K_1 \subseteq K_2$?

Due to the necessary uncertainty around the boundary, in both cases we can only expect an approximate answer; we specify an $\varepsilon > 0$ and if $\text{vol}(K_1 \cap K_2) < \varepsilon$, or if $\text{vol}(K_1 \setminus K_2) < \varepsilon$, then the "YES" answer is acceptable.

Having separation oracles for $K_1$ and $K_2$, we can design a separation algorithm for $K_1 \cap K_2$ trivially. Suppressing technical details (the number $r$), the Equivalence Principle yields a polynomial time solution for the violation problem for $K_1 \cap K_2$.

The second problem is more difficult; in fact, to solve it (even with the $\varepsilon$-tolerance) would yield polynomial-time solutions for NP-complete problems. But using the randomized methods discussed in Chapter 4 below, we can test $K_1 \subset K_2$ (with the $\varepsilon$-tolerance) in time polynomial in $n$ and $1/\varepsilon$. (Note that a "truly" polynomial algorithm ought to be polynomial in $\log(1/\varepsilon)$.)

The most interesting applications of the Equivalence Principle are in the field of combinatorial optimization. We illustrate this by two results due to Grötschel, Lovász and Schrijver (see 1988). Consider the problem of determining $\alpha(G)$ for a t-perfect graph $G$. As discussed earlier, this is equivalent to maximizing the linear objective function $\sum x_i$ over $\text{STAB}(G)$, which could be easily solved by binary search if we could test validity for $\text{STAB}(G)$. By the Equivalence Principle, it would suffice to find a polynomial time membership test for $\text{STAB}(G)$. In other words, given a vector $x \in \mathbb{R}^V$, is $x \in \text{STAB}(G)$?

At this point we want to use the linear description of $\text{STAB}(G)$, which we have determined for t-perfect graphs: $\text{STAB}(G)$ is the solution set of inequalities (1)–(2)–(3) above. So all we have to do is to test whether $x$ satisfies these inequalities.
For (1) and (2), this is trivially checked by simply substituting $x$ in them. But for (3) this does not work: there are (typically) exponentially many inequalities in (3), and we do not want to generate them all.

However, there is a way to check in polynomial time whether or not $x$ satisfies all these inequalities. We may assume that we have checked (1) and (2) and that $x$ satisfies these. For every edge $ij \in E$, let $y_{ij} = 1 - x_i - x_j$. Since $x$ satisfies (2), these numbers are non-negative, and we may consider them as “lengths” of the edges. Let $C$ be an odd circuit in $G$, then

$$
\sum_{i \in E(C)} y_{ij} = |C| - \sum_{i \in V(C)} x_i,
$$

and hence $x$ satisfies the constraint of (3) belonging to this circuit $C$ if and only if the “length” of $C$ is at least 1. So $x$ satisfies (3) iff the length of every odd circuit is at least 1.

Now there is a rather simple version of breadth-first-search that finds the shortest odd circuit in any graph with non-negative edge-lengths (the details of this do not belong here). Using this, all we have to do is to check whether this minimum length is less than 1 or not.

The stability number of a perfect graph can also be determined in polynomial time. This algorithm is also based on the Equivalence Principle, but it is substantially more complicated. If we want to copy the above argument, we run into the following difficulty: the membership test for $\text{STAB}(G)$ is equivalent to the validity test for the stable set polytope of the complementary graph, and so the equivalence principle reduces the problem of determining a maximum stable set to (essentially) the same problem for the complementary graph. One has to go up to the $|V|^2$ dimensional space to apply the Equivalence Principle successfully.

This type of application of the Equivalence Principle is not rare; in fact, it is shown by Grötschel, Lovász and Schrijver (see 1988) that most of those combinatorial optimization problems whose polynomial time solvability was known by (often quite involved) ad hoc algorithms, can be solved in polynomial time using a combination of the Equivalence Principle with very elementary algorithmic ideas. There are also several combinatorial optimization problems which can be solved in polynomial time, but for which no polynomial time algorithm avoiding the use of the Equivalence Principle is known (two very important such problems are finding the independence number of a perfect graph and minimizing a submodular setfunction).

Unfortunately, algorithms derived from the Equivalence Principle are polynomial, but very slow (their running time is a polynomial with a very high degree), and therefore practically useless. This is quite natural, considering how general this method is. These results should be interpreted as “existence proofs” for polynomial time solvability of the problem; having established this much, one can try to design problem-specific algorithms which are more efficient. The remarks above show that this is often very difficult.
3. Convex Bodies and Ellipsoids

3.1 The Löwner-John Ellipsoid

It was proved by Löwner that for each convex body \( K \), there exists a unique ellipsoid \( E \) with minimum volume containing it. John proved that if we shrink this ellipsoid of a convex body \( K \) from its center by a factor of \( n \), we obtain an ellipsoid that is contained in \( K \). We call this ellipsoid the Löwner-John ellipsoid of the body (see (Grötschel, Lovász and Schrijver 1988) for details). Analogues of the theorems above hold if we consider an inscribed ellipsoid with largest volume.

If we restrict ourselves to centrally symmetric convex bodies, then of course the center will be the center of the Löwner-John ellipsoid as well, and in John's theorem it suffices to shrink by a factor of \( \sqrt{n} \).

The Löwner-John ellipsoid itself may be difficult to compute. However, an ellipsoid \( E \) with somewhat weaker properties can be computed in polynomial time, using a version of the shallow cut ellipsoid method due to Yudin and Nemirovskii (1976). We call an ellipsoid \( E \) a weak Löwner-John ellipsoid for \( K \), if \( E \) contains \( K \) and if we shrink \( E \) from its center by a factor of \( 2n^{3/2} \), we obtain an ellipsoid that is contained in \( K \).

**Theorem 3.1** Given a separation oracle for a convex body \( K \), a weak Löwner-John ellipsoid for \( K \) can be computed in polynomial time.

We remark that if \( K \) is given in a more explicit manner (e.g., as the solution set of a system of linear inequalities or the convex hull of a set of vertices) then the factor \( n^{3/2} \) can be improved to \( 2n \).

The algorithm proving Theorem 3.1 is based on a natural proof of John's theorem. Let \( E \) be the ellipsoid with smallest volume containing \( K \). Applying an affine transformation, we may assume that \( E \) is the unit ball about the origin. Let \( E' \) denote the ball with radius \( 1/n \) about the origin, and assume that \( E' \) is not contained in \( K \). Let \( x \in E' \setminus K \), and let \( H \) be a hyperplane separating \( x \) from \( K \). \( H \) cuts \( E \) into two parts, one of which, say \( E_1 \), includes \( K \). Now \( E_1 \) is smaller or only slightly larger than a half-ball, since \( x \not\in E_1 \). Therefore it is a routine computation in linear algebra to verify that \( E_1 \) can be included in an ellipsoid which has smaller volume than \( E \).

To turn this proof into an algorithm, two ideas have to be added. We want to start out with some ellipsoid containing \( K \) (say, with the ball with radius \( R \) about the origin), and replace it as in the above proof with ellipsoids of smaller and smaller volume until a weak Löwner-John ellipsoid is obtained. In order to guarantee a good running time, we have to reduce the volume by a substantial factor like \( 1 - (1/n^2)^2 \); this will be achieved if the radius of \( E' \) is chosen, say, \( 1/2n \) instead of \( 1/n \). The other problem is to find the vector \( x \in E' \setminus K \). This is easy if \( K \) is the solution set of a system of linear inequalities, but hard in general. What we can do is to test the \( 2n \) intersection points of the axes with the surface of \( E' \) for membership in \( K \); if one of them is not in \( K \), then the separation algorithm yields the hyperplane \( H \). If each of them is in \( K \) then \( K \) contains the smaller ball obtained from \( E' \) by shrinking by a factor of \( \sqrt{n} \). Thus \( E \) is a weak Löwner-John ellipsoid.

Recently some interesting new results concerning the computation of the "true" Löwner-John ellipsoid have been obtained. If \( K \) is a convex polytope presented by an explicit list of its vertices, then for any \( \epsilon > 0 \) an approximation
of the Löwner-John ellipsoid with error at most $\varepsilon$ can be found in polynomial time. The "polar" problem of finding an inscribed ellipsoid with (approximately) maximum volume can also be solved in polynomial time, provided the polytope is presented as the solution set of an explicit system of linear inequalities. Such an algorithm was given by Nesterov and Nemirovskii (1989), and improved by Khachiyan and Todd (1990). These results are very presentation-dependent; Khachiyan and Todd conjecture that to compute an approximate Löwner-John ellipsoid for a polytope presented by linear inequalities is NP-hard.

The diameter and width of the weak Löwner-John ellipsoid can be used to estimate the diameter and width of $K$ with relative error at most $n^{3/2}$. This estimate is not as bad as it looks at a first glance; it can be shown (Bárány and Füredi 1986) that if $K$ is given by any of the oracles above, then any estimate on the width of a body $K$ computable in polynomial time is off by a factor which grows as a power of $n$.

3.2 The Ellipsoid Method

The algorithm sketched above is the heart of various versions of the algorithm called the Ellipsoid Method, developed by Shor (1970) and Yudin and Nemirovskii (1976). The method became widely known when Khachiyan (1979) applied it to obtain the first polynomial time linear programming algorithm. To show the idea, assume that we want to find the optimum of a linear objective function $c \cdot x$ over a convex body $K \subseteq \mathbb{R}^n$ for which we have a separation oracle. Design a black box which does the following: we plug in a vector $x_0 \in \mathbb{R}^n$. If $x_0 \not\in K$, it returns a hyperplane separating $x$ from $K$. If $x_0 \in K$, it returns the hyperplane $c - x = c - x_0$. This is essentially a separation oracle for the set of vectors in $K$ optimizing the linear objective function $c \cdot x$. Even though this convex set is not full-dimensional, an appropriate modification of the argument above gives that we can use this black box as a separation oracle in the ellipsoid method and obtain an "approximately optimal" point in $K$.

Once we can solve the optimization problem over $K$, the validity and violation problems are easily settled.

It is more difficult to handle these problems when $K$ is given by a membership test rather than by a separation oracle. It takes a nice but difficult argument by Yudin and Nemirovskii (1976) to settle this case.

3.3 Karmarkar's Method

The ellipsoid method yields a polynomial time algorithm to solve linear programs, but it is too slow in practice to be useful in actual computations. The classical method to solve linear programs, namely Dantzig's simplex method, takes exponential time in the worst case but works very efficiently for the majority of real-life problems. A linear programming algorithm which is both theoretically efficient (polynomial-time) and competitive with the simplex method in practice was given by Karmarkar (1984). Since this method also has a very geometric background, it is worth sketching here.

Assume that we want to maximize a linear objective function $c \cdot x$ over a convex body $K$. Also assume that we have already found an interior point $x_0$ and we want to find a sequence of interior points $x_1, x_2, \ldots$ converging to the
optimum point. When stepping from \( x_i \) to \( x_{i+1} \), it would be natural to move in the direction of \( c \), or, in other words, orthogonal to the hyperplane \( c \cdot x = \text{const} \). This leads us, however, to points close to the boundary and (possibly) far from the optimum, and then we have to move in some other direction.

However, it is not really natural to move orthogonal to \( c \); this is not invariant under affine transformations while the hole task is. The idea of many optimization algorithms is to change the metric of the space so that the notion of "orthogonality" should be tailored to the particular body (such are the variable metric methods, but the ellipsoid method can also be viewed this way).

There is a very natural metric inside every convex body, introduced by Hilbert. The distance of any two points \( u, v \in \text{int}(K) \) is defined as the logarithm of the cross ratio of \( u, v \), and the two points of intersection of the line through \( u \) and \( v \) with the boundary of \( K \). (This construction may also be familiar from the Caley-Klein model of hyperbolic geometries.) It is not difficult to see that if we move in the direction of steepest descent with respect to this metric to, say, half way to the boundary, and then repeat this, then we get very close to the optimum very fast. For example, if \( K \) is centrally symmetric with respect to \( x_0 \) then we approach the optimum on a straight line and the distance from the optimum is halved at each step.

Unfortunately, there is no easy way to compute the Hilbert metric in a convex body, but we can approximate it in the neighborhood of an interior point \( v \) by finding a projective transformation which maps \( K \) onto a convex body \( K' \) so that the unit ball includes \( K \), \( K \) includes the ball with a reasonably large radius \( q \) about the origin, and \( v \) is mapped onto the origin. Then the euclidean metric will approximate the projective metric. Except for the condition on \( v \), this sounds like the ellipsoid method, but we have more freedom because we are allowed to use projective transformations. (The optimization problem itself is not projective invariant, but one can transform it into projective invariant problems easily; for example, the Violation Problem above is projective invariant.)

The crucial part of Karmarkar's method is to construct such a projective transformation very fast in the case of polytopes presented by an explicit system of \( m \) linear inequalities. The inner radius \( q \) he achieves is \( 1/m \); this shows that (unlike the ellipsoid method) this method is sensitive to the number of constraints. This is of course only a very rough sketch; many ingenious details must be added to make this procedure work for really large linear programs. For details, extensions, and related algorithms see the survey of Todd (1989).

4. The Volume of a Convex Body

Now we turn to the fundamental problem of determining, or at least estimating, the volume of a convex body. This question has recently brought exciting developments. For a while, a number of negative results were obtained (Section 4.1), which showed that even to compute an estimate in polynomial time with decent relative error is hopeless. But recently Dyer, Frieze and Kannan (1989) designed a randomized polynomial time algorithm (i.e., an algorithm making use of a random number generator) which computes an estimate of the volume such that the probability that the relative error is larger than a prescribed \( \varepsilon > 0 \) is arbitrarily small. This outstanding result uses a number of tools from probability and geometry, and we sketch it in Section 4.2.
4.1 The Difficulty of Computing the Volume

One thing we can do is to find a weak Löwner-John ellipsoid of the body. Since it is easy to follow how an affine transformation modifies the volume (it multiplies by the determinant of the corresponding matrix), we may assume that the body is contained in the unit ball and contains the ball with radius $1/2n^{3/2}$ about the origin. In this case, we have the trivial bounds on the volume:

$$\frac{\pi^{n/2}}{2^{n/2}n^{3n/2}} \leq \text{vol}(K) \leq \frac{\pi^{n/2}}{\Gamma(1 + n/2)}.$$

So for an arbitrary convex body (given by, say, a separation oracle), we can compute an upper bound on its volume with relative error at most $2^{n}n^{3n/2}$.

The following surprising result of Bárány and Füredi (1986), improving a somewhat weaker bound given by Elekes (1986), shows that no substantially better estimate can be given on the volume, at least if the convex body is given by one of the four equivalent oracles discussed in Chapter 2.

**Theorem 4.1.** Consider any polynomial time algorithm which assigns to every convex body $K$, given by (say) a membership oracle, an upper bound $w(K)$ on $\text{vol}(K)$. Then there is a constant $c > 0$ such that in every dimension $n$ there exists a body $K$ for which $w(K) > n^{cn}\text{vol}(K)$.

Let us sketch the proof of this result. It depends on the following geometric lemma:

**Lemma 4.2.** There exists a constant $c > 0$ such that the volume of the convex hull of any $p > 0$ points in the unit ball is less than $p \cdot n^{-cn}$.

To prove Theorem 4.1, first apply the algorithm to the unit ball $B$. The algorithm runs and asks the membership of a polynomial number $p$ of points $v_1, \ldots, v_p$ from the oracle, and computes an upper bound $w(B)$ of the volume. Now apply the algorithm to the convex hull $K$ of $\{e_1, \ldots, e_n, v_1, \ldots, v_p\} \cap B$. It follows that the algorithm runs exactly as in the previous case: it asks the same questions from the oracle and therefore it gets the same answers. So the algorithm finds the same estimate $w(K) = w(B)$. But since the volume of $K$ is much smaller than the volume of $B$, this estimate must have a large relative error.

This result does not say anything about computing the volume of convex bodies given in any specific way, say as the solution set of a system of linear inequalities. However, in both cases the exact computation of the volume is NP-hard (Dyer and Frieze 1988, Khachiyan 1988) and even #P-hard (Khachiyan 1989). (This latter fact also follows from the result of Brightwell and Winkler (1990) mentioned in Section 1.3.)

The approximate computation of the volume of explicitly described polyhedra is an open problem.
4.2 Markov Chains, Isoperimetric Inequalities, and Approximating the Volume

In this section we sketch the randomized algorithm of Dyer, Frieze and Kannan (and variants) for estimating the volume of a convex body $K \subseteq \mathbb{R}^n$. About this body, we only need to assume that it is given by a separation oracle.

At this point, many readers may ask: what’s wrong with the straightforward Monte-Carlo algorithm? We have already made the assumption that $K$ is included in the unit ball $B$. Let us generate many random points in $B$, and count how often we hit $K$. This gives us an estimate on the ratio of the volumes of $K$ and $B$.

The problem is that the volume of $K$ may be smaller than the volume of $B$ by an exponential factor (in $n$). Hence the first exponentially many random points will miss the body $K$. This method can be applied to estimate the ratio of the volumes of two convex bodies (one including the other) only if this ratio is not too small.

This suggests the first trick: “connect” $K$ and $B$ by a sequence of convex bodies $K = K_0 \subseteq K_1 \subseteq \ldots \subseteq K_m = B$, so that $\text{vol}(K_i)/\text{vol}(K_{i+1}) \geq 1/2$ (and $m$ is polynomial in $n$). Then these ratios can be estimated by the Monte-Carlo method, and their product gives an estimate on the ratio $\text{vol}(K)/\text{vol}(B)$. Such a sequence is easily constructed: we can take e.g. $K_i = B \cap (1 + \frac{1}{2n})^i K$. Since $K$ contains the ball $(1/(2n^{3/2})) K$, it follows that $K_i = B$ for $i \geq 4n \log n$.

However, estimating $\text{vol}(K_i)/\text{vol}(K_{i+1})$ by the Monte-Carlo method is not so easy; the key question in this algorithm (and in all versions of it) is:

How to generate a random point (with uniform distribution) in a convex body?

The solution by Dyer, Frieze and Kannan is the following. Consider the lattice of vectors whose coordinates are integer multiples of a sufficiently small number $\delta$. Call two lattice points adjacent if their distance is exactly $\delta$. Starting from the origin, take a random walk on the lattice points in $K$. If we are at a lattice point $v$, select an adjacent lattice point $w$ at random. If $w \in K$, then move to $w$; else, stay at $v$. (For technical reasons, we flip a coin before the move and if it falls on head, we stay where we are anyway.) After an appropriate number of steps, we stop; our current position can be considered as a random point in $K$.

It is easy to see, using the theory of Markov chains, that if $w_t$ denotes the (random) lattice point obtained after $t$ steps, then the distribution of $w_t$ tends to the uniform distribution over the set $V$ of lattice points accessible from the origin along legal walks. (Note that $V$ is essentially $L \cap K$, except possibly for some lattice points near the boundary of $K$.)

The problem is to find a good bound on the rate of this convergence (on the mixing rate of the Markov chain $(w_0, w_1, \ldots)$). By general results on the mixing rate of Markov chains (Sinclair and Jerrum 1989), we know that this depends on the conductance of the Markov chain. This (in our case) can be defined as the largest $\Phi > 0$ such that for every $S \subseteq V$, the number of pairs $(u, v)$ of adjacent lattice points with $u \in S$ and $v \in V \setminus S$ is at least $\Phi \cdot (2n) \cdot \min\{|S|, |V \setminus S|\}$. The least $t$ for which the distribution $w_t$ is essentially uniformly distributed is about $1/\Phi^2$.

So the question is to find a lower bound on the conductance of this Markov chain. Let $S \subseteq V$ and let $K_1$ be the set of points in $K$ nearer to $S$ than to $V \setminus S$. Then we expect that
(a) the volume of $K_1$ is about $\delta^n |S|$,  
(b) the volume of $K \setminus K_1$ is about $\delta^n |V \setminus S|$,  
(c) the surface area of the $K_1$ inside $K$ is about $\delta^{n-1}$ times the number of adjacent pairs of lattice points $(u, v)$ with $u \in S$ and $v \in V \setminus S$.

If we accept these approximations (which is a technically quite difficult part of the argument), then the problem reduces to the following isoperimetric inequality:

**Theorem 4.3.** Let $K$ be a convex body in $\mathbb{R}^n$ with diameter $d$. Let $F$ be a surface with $(n-1)$-dimensional measure $f$, cutting $K$ into two parts with volumes $v_1$ and $v_2$. Then

$$ f \geq \frac{1}{d} \min\{v_1, v_2\}. $$

A weaker inequality (but still sufficient to prove the polynomiality of the volume algorithm) was proved by Dyer, Frieze and Kannan. This form is due to Karzanov and Khachiyan (1990) (who use methods from differential geometry) and to Lovász and Simonovits (1990) (who use the so-called Ham-Sandwich Theorem; to justify the remark that algebraic topology also plays some role, let us mention that the Ham-Sandwich Theorem is derived from the Brouwer Fixed Point Theorem).

Unfortunately, even with these improvements the algorithm is practically useless; its running time grows with the 16th power of $n$. The main problem is that random walks are slow in getting to the distant part of $K$. It is easy to come up with other Markov chains of points in $K$ which “jump around” faster (and also have uniform limit distribution). Unfortunately, to prove the corresponding analogues of the isoperimetric inequality in Theorem 4.3 seems to take new methods. It is clear the algorithm of Dyer, Frieze and Kannan, representing a theoretical breakthrough, is not the last word in the area of volume algorithms.

An algorithm to efficiently generate random points in a convex body has many further applications. Numerical integration by Monte-Carlo methods is an obvious one. We have also mentioned in Section 2.2 the problem to test whether a convex body $K_1$ is included in another convex body $K_2$. This can be done by generating many random points in $K_1$ and test whether these are contained in $K_2$. (To be convinced that at most a fraction of $\varepsilon$ of the volume of $K_1$ is not contained in $K_2$, we have to generate about $1/\varepsilon$ points. Thus this algorithm is not “truly” polynomial if $\varepsilon$ is part of the input.)

**References**


