Steffen Rebennack
Ellipsoid Method
in *Encyclopedia of Optimization*, Second Edition,
C.A. Floudas and P.M. Pardalos (Eds.),
Springer, pp. 890–899, 2008

Ellipsoid Method

Steffen Rebennack
University of Florida
Center of Applied Optimization
Gainesville, FL 32611, USA
e-mail: steffen@ufl.edu

July 2007

Key words: Ellipsoid Method, linear programming, polynomially solvable, linear inequalities, separation problem

In this article we give an overview of the Ellipsoid Method. We start with a historic introduction and provide a basic algorithm in Section 2. Techniques to avoid two important assumptions required by this algorithm are considered in Section 2.2. After the discussion of some implementation aspects, we are able to show the polynomial running time of the Ellipsoid Method. The second section is closed with some modifications in order to speed up the running time of the ellipsoid algorithm. In Section 3, we discuss some theoretical implications of the Ellipsoid Method to linear programming and combinatorial optimization.

1 Introduction

In 1979, the Russian mathematician Leonid G. Khachiyan published his famous paper with the title "*A Polynomial Algorithm in Linear Programming*", [Kha79]. He was able to show that linear programs (LPs) can be solved efficiently; more precisely that LP belongs to the class of polynomially solvable problems. Khachiyan's approach was based on ideas similar to the Ellipsoid Method arising from convex optimization. These methods were developed by David Yudin and Arkadi Nemirovski, [YN76a, YN76b, YN77], and independently by Naum Shor, [Sho77], preceded by other methods as for instance the Relaxation Method, Subgradient Method or the Method of Central Sections, [BGT81]. Khachiyan's effort was to modify existing methods enabling him to prove the polynomial running time of his proposed algorithm. For his work, he was awarded with the

Fulkerson Prize of the American Mathematical Society and the Mathematical Programming Society, [MT05, Sia05].

KHACHIYAN's four-page note did not contain proofs and was published in the journal Soviet Mathematics Doklady in February 1979 in Russian language. At this time he was 27 years young and quite unknown. So it is not surprising that it took until the Montreal Mathematical Programming Symposium in August 1979 until KHACHIYAN's breakthrough was discovered by the mathematical world and a real flood of publications followed in the next months, [Tee80]. In the same year, *The New York Times* made it front-page news with the title "*A Soviet Discovery Rocks World of Mathematics*". In October 1979, the *Guardian* titled "*Soviet Answer to Traveling Salesmen*" claiming that the Traveling Salesman problem has been solved – based on a fatal misinterpretation of a previous article. For an amusing outline of the interpretation of KHACHIYAN's work in the world press, refer to [Law80].

2 Method

The Ellipsoid Method is designed to solve decision problems rather than optimization problems. Therefore, we first consider the decision problem of finding a feasible point to a system of linear inequalities

$$A^\top x \leq b \tag{1}$$

where A is a $n \times m$ matrix and b is an n-dimensional vector. From now on we assume all data to be integral and n to be greater or equal than 2. The goal is to find a vector $x \in \mathbb{R}^n$ satisfying (1) or to prove that no such x exists. We see in Section 3.1 that this problem is equivalent to a linear programming optimization problem of the form

$$\begin{aligned} \min_x \quad & c^\top x \\ s.t. \quad & A^\top x \leq b, \\ & x \geq 0, \end{aligned}$$

in the sense that any algorithm solving one of the two problems in polynomial time can be modified to solve the other problem in polynomial time.

2.1 The Basic Ellipsoid Algorithm

Roughly speaking, the basic idea of the Ellipsoid Method is to start with an initial ellipsoid containing the solution set of (1). The center of the ellipsoid is in each step a candidate for a feasible point of the problem. After checking whether this point satisfies all linear inequalities, one either

produced a feasible point and the algorithm terminates, or one found a violated inequality. This is used to construct a new ellipsoid of smaller volume and with a different center. Now the procedure is repeated until either a feasible point is found or a maximum number of iterations is reached. In the latter case, this implies that the inequality set has no feasible point.

Let us now consider the presentation of ellipsoids. It is well known, that the n-dimensional ellipsoid with center x^0 and semi-axis g_i along the coordinate axis is defined as the set of vectors satisfying the equality

$$\sum_{i=1}^{n} \frac{(x_i - x_i^0)^2}{g_i^2} = 1. \tag{2}$$

More general, we can formulate an ellipsoid algebraically as the set

$$E := \{x \in \mathbb{R}^n \,|\, (x - x^0)^\top B^{-1}(x - x^0) = 1\}, \tag{3}$$

with symmetric, positive definite, real-valued $n \times n$ matrix B. This can be seen with the following argument. As matrix B is symmetric and real-valued, it can be diagonalized with a quadratic matrix Q, giving $D = Q^{-1}BQ$, or equivalently, $B = QDQ^{-1}$. The entries of D are the eigenvalues of matrix B, which are positive and real-valued. They will be the quadratic, reciprocal values of the semi-axis g_i. Inserting the relationship for B into (3) yields to $(x - x^0)^\top QD^{-1}Q^{-1}(x - x^0) = 1$ which is equivalent to

$$\left((x - x^0)^\top Q\right) D^{-1} \left((x - x^0)^\top Q\right)^\top = 1.$$

Hence, we can interpret matrix Q as a coordinate transform to the canonical case where the semi-axis of the ellipse are along the coordinate axis. Recognize that in the case when matrix B is a multiple of the unit matrix, $B = r^2 \cdot I$, then the ellipsoid gets a sphere with radius $r > 0$ and center x^0. We abbreviate this in the following by $S(x^0, r)$.

We start with a somewhat basic version of the ellipsoid algorithm. This method requires two important assumptions on the polyhedron

$$P := \{x \in \mathbb{R}^n \,|\, Ax \leq b\}. \tag{4}$$

We assume that

1. the polyhedron P is bounded and that
2. P is either empty or full-dimensional.

In Section 2.2, we will see how this algorithm can be modified not needing these assumptions. This will allow us to conclude that a system of linear inequalities can be solved in polynomial running time with the Ellipsoid Method.

Let us now discuss some consequences of these two assumptions. The first assumption allows us to construct a sphere $S(c_0, R)$ with center c_0 and radius R containing P completely: $P \subseteq S(c_0, R)$. The sphere $S(c_0, R)$ can be constructed, for instance, in the following two ways. If we know the bounds on all variables x, e.g. $L_i \leq x_i \leq U_i$, one can use a geometric argument to see that with

$$R := \sqrt{\sum_{i=1}^{n} \max\{|L_i|, |U_i|\}^2},$$

the sphere $S(0, R)$ will contain the polytope P completely. In general, when such bounds are not given explicitly, one can use the integrality of the data and proof, see for instance [GLS88], that the sphere with center 0 and radius

$$R := \sqrt{n} 2^{<A>+<b>-n^2} \tag{5}$$

contains P completely, where $< \cdot >$ denotes the encoding length of some integral data. For an integer number b_i, we define

$$< b_i >:= 1 + \lceil \log_2(|b_i| + 1) \rceil ,$$

which is the number of bits needed to encode integer b_i in binary form; one bit for the sign and $\lceil \log_2(|b_i| + 1) \rceil$ bits to encode $|b_i|$. With this, the encoding length of a vector b is the sum of the encoding lengths of its components. Similarly, for a matrix A, the encoding length is given by $< A >:= \sum_{i=1}^{m} < a_i >$.

The second assumption implies that if P is non-empty, its volume is strictly positive, meaning that there is an n-dimensional sphere of radius $r > 0$ which is contained in P. More precisely, it is possible to show that

$$vol(P) \geq 2^{-(n+1)<A>+n^3}, \tag{6}$$

in the case that P is not empty, see [GLS81]. This will help us to bound the number of iterations of the basic ellipsoid algorithm. The graphical interpretation of the positive volume of polytope P is that the solution set of system (1) is not allowed to have mass zero, for instance, not to be a hyperplane.

Figure 1 illustrates the effect of the two assumptions in the case that P is non-empty. As this is a two-dimensional example, the second assumption implies that polytope P is not just a line segment but instead contains a sphere of positive radius r.

Now, we are ready to discuss the main steps of the basic ellipsoid algorithm. Consider therefore Algorithm 2.1. In the first six steps the algorithm is initialized. The first ellipsoid is defined in step three. The meaning of the parameters for the ellipsoid and especially number k^* will be discussed later in this section. For now, let us also ignore step seven. Then, for each iteration k, it is checked in step eight if center x^k satisfies the linear inequality system (1). This can be done for instance

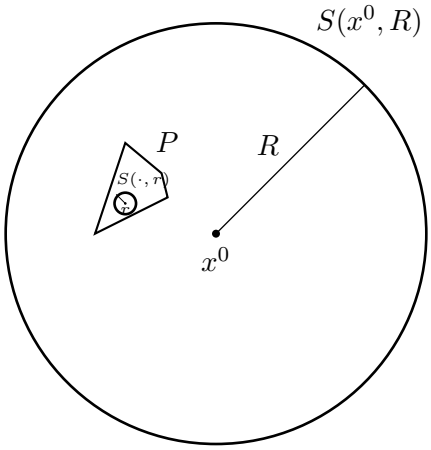


Figure 1: P is bounded and full-dimesional

by checking each of the m inequalities explicitly. In the case that all inequalities are satisfied, x^k is a feasible point and the algorithm terminates. In the other case there is an inequality $a_j^\top x \leq b_j$ which is violated by x^k, step nine. In the next two steps, a new ellipsoid E_{k+1} is constructed. This ellipsoid has the following properties. It contains the half ellipsoid

$$H := E_k \cap \{x \in \mathbb{R}^n \,|\, a_j^\top x \leq a_j^\top x^k\} \tag{7}$$

which insures that the new ellipsoid E_{k+1} also contains polytope P completely, assuming that the initial ellipsoid $S(0, R)$ contained P. Furthermore, the new ellipsoid has the smallest volume of all ellipsoids satisfying (7), see [BGT81]. The central key for the proof of polynomial running time of the basic ellipsoid algorithm 2.1 is another property, the so called *Ellipsoid Property*. It provides the following formula about the ratio of the volumes of the ellipsoids

$$\frac{volE_{k+1}}{volE_k} = \sqrt{\left(\frac{n}{n+1}\right)^{n+1}\left(\frac{n}{n-1}\right)^{n-1}} < \exp(-\frac{1}{2n}), \tag{8}$$

for a proof see, for instance, [GLS81, NW99]. As $\exp(-1/(2n)) < 1$ for all natural numbers n, the new ellipsoid has a strict smaller volume than the previous one. We also notice that $\exp(-1/(2n))$ is a strictly increasing function in n which has the consequence that the ratio of the volumes of the ellipsoids is closer to one when the dimension of the problem increases.

Now, let us discuss the situation when P is empty. In this case, we want Algorithm 2.1 to terminate in step seven. To do so, we will derive an upper bound k^* on the number of iterations needed to find a center x^k satisfying the given system of linear inequalities for the case that P is not empty. Clearly, if the algorithm would need more than k^* iterations, the polytope P must be empty. Therefore, let us assume again that $P \neq \emptyset$. In this case (6) provides a lower bound for the volume of P and an upper bound of its volume is given, for instance, by (5). In addition, according to the construction of the ellipsoids, we know that each of the E_{k+1} contains P completely. Together with the Ellipsoid Property (8) we get the relation

$$vol(E_{k^*}) < \exp(\frac{-k^*}{2n})vol(E_0) < 2^{\frac{k^*}{2n}+n+n\log(R)} \stackrel{!}{=} 2^{-(n+1)<C>+n^3} \leq vol(P).$$

This chain provides an equation defining the maximum number of iterations

$$k^* := 2n(2n+1) < C > +2n^2(\log(R) - n^2 + 1). \tag{9}$$

A geometric interpretation is that the volume of the ellipsoid in the k^*th iteration would be too small to contain the polytope P. Obviously, this implies that P has to be empty. With this, we have shown that the presented basic ellipsoid algorithm works correctly.

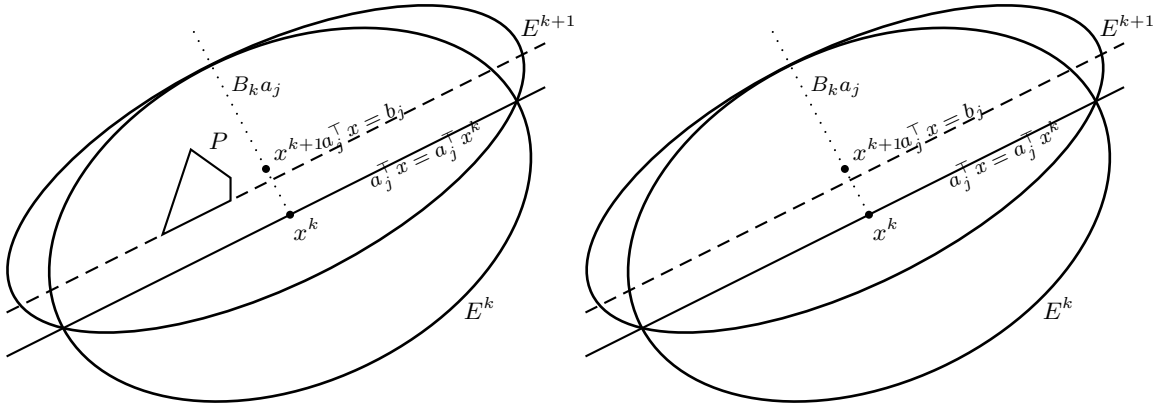
Algorithm 2.1 Basic Ellipsoid Algorithm

Input: Matrix A, vector b; sphere $S(c_0, R)$ containing P
Output: feasible $\overline{x}$ or proof that $P = \emptyset$

// Initialize
1: $k := 0$
2: $k^* = 2n(2n+1) < C > +2n^2(\log(R) - n^2 + 1)$ // Max number of iterations
3: $x^0 = c_0$, $B_0 = R^2 \cdot I$ // Initial ellipsoid
4: $\tau := \frac{1}{n+1}$ // Parameter for ellipsoid: step
5: $\sigma := \frac{2}{n+1}$ // Parameter for ellipsoid: dilation
6: $\delta := \frac{n^2}{n^2-1}$ // Parameter for ellipsoid: expansion
// Check if polytope P is empty
7: **if** $k = k^*$ **return** $P = \emptyset$
// Check feasibility
8: **if** $A\,x^k \leq b$ **return** $\overline{x} := x^k$ // $\overline{x}$ is a feasible point
9: **else** let $a_j^\top x^k > b_j$
// Construct new ellipsoid
10: $B_{k+1} := \delta(B_k - \sigma\frac{B_k a_j (B_k a_j)^\top}{a_j^\top B_k a_j})$
11: $x^{k+1} := x^k - \tau\frac{B_k a_j}{\sqrt{a_j^\top B_k a_j}}$
// Loop
12: $k \leftarrow k+1$ and goto step 7

One iteration of the basic ellipsoid algorithm, for the case of a non-empty polytope P, is illustrated in Figure 2 (a). We recognize that P is contained completely in E_k. The dashed line shows equality $a_j^\top x \leq b_j$ corresponding to one of the two inequalities which are violated by x^k. Geometrically, this equality is moved in parallel until it contains center x^k. Recognize that the new ellipsoid E_{k+1} contains the half ellipsoid (7). In this case, the new center x_{k+1} is again not contained in polytope P and at least one more step is required. The case that polytope P is empty is illustrated in Figure 2 (b) which is mainly the same as Figure 2 (a).

In the case that B_k is a multiple of the identity, the ellipsoid is an n-dimensional sphere. According to the initialization of the basic ellipsoid algorithm in step three, this is the case for the first iteration when $k = 0$. This gives us an interpretation of the values of δ and τ. The new ellipsoid E_{k+1} is shrunk by the factor $\sqrt{\delta(1-\sigma)} = n/(n+1)$ in the direction of vector a_j and expanded



(a) $P \neq \emptyset$ (b) $P = \emptyset$

Figure 2: Basic ellipsoid algorithm

in all orthogonal directions by factor $\sqrt{\delta} = n/\sqrt{n^2-1}$. Hence, we see that in the next step we do no longer get a sphere if we start with one. The third parameter, the step, gives intuitively the measure how far we go from point x^k in the direction of vector $B_k a_j$, multiplied by factor $1/\sqrt{a_j^\top B_k a_j}$. For more details we refer to the survey of Bland, Goldfarb and Todd, [BGT81].

After all the discussions above, we are now able to conclude the polynomial running time of the basic ellipsoid algorithm 2.1 for the case that the polyhedron P is bounded and either empty or full-dimensional. For an algorithm to have polynomial running time in the deterministic Turing-machine concept, there has to be a polynomial in the encoding length of the input data describing the number of elementary steps needed to solve an arbitrary instance of the problem. Therefore, let $L := < A, b >$ be the encoding length of the input data. Obviously, each of the steps of Algorithm 2.1 can be done in polynomial many steps in L. The maximum number of iterations is given through k^* which is also a polynomial in L; consider therefore equations (9) and (5). Hence, we conclude that the ellipsoid algorithm 2.1 has a polynomial running time. During the reasoning above, we required implicitly one more important assumption, namely to have exact arithmetic. Normally, this is of more interest in numerics rather than in theoretical running time analysis. However, it turns out that for the Ellipsoid Method this is crucial as the Turing-machine concept also requires finite precision. Let us postpone this topic to the end of the next subsection and rather discuss methods to avoid the two major assumptions on polyhedron P.

2.2 Polynomially running time: Avoiding the assumptions

In order to prove that the Ellipsoid Method can solve the system of linear inequalities (1) in polynomial time, one has to generalize the basic ellipsoid algorithm 2.1 to need not the assumptions that (1) polyhedron P is bounded, (2) P is either empty or full-dimensional and (3) that exact arithmetic is necessary.

In 1980, KHACHIYAN published a paper, [Kha80], discussing all the details and proofs about the Ellipsoid Method which were neglected in his paper from 1979. In Lemma 1, he showed that

$$P \cap S(0, 2^L) \neq \emptyset, \tag{10}$$

in the case that $P \neq \emptyset$. With this, one can use for R of (5) value 2^L. However, we cannot just adopt the algorithm above to the new situation. Instead of a lower bound for $vol(P)$, as given in (6), we would need a lower bound for $vol(P \cap S(0, 2^L))$. To achieve this, we follow a trick introduced by KHACHIYAN and consider the perturbed system of linear inequalities

$$2^L a_i^\top x \leq 2^L \beta_i + 1 \qquad i = 1, \ldots, m. \tag{11}$$

Let us abbreviate the corresponding solution set with P', which is in general a polyhedron. KHACHIYAN was able to proof a one-to-one correspondence of the original system (1) and the perturbed one (11). This means that P is empty if and only if P' is empty, it is possible to construct a feasible point for (1) out of (11) in polynomial time and the formulation of the perturbed formulation is polynomial in L. Furthermore, the new inequality system has the additional property that if $P' \neq \emptyset$ it is full-dimensional. Hence, it is possible to find a (non-empty) sphere included in P'. It can be shown, that

$$S(\overline{x}, 2^{-2L}) \subseteq P' \cap S(0, 2^L),$$

where $\overline{x}$ is any feasible point of the original system (1) and hence $\overline{x} \in P$. With this argument at hand, it is possible to derive an upper bound for the number of iterations for the Ellipsoid Method by solving the perturbed system (11). It can be shown that a feasible point can be found in at most $6n(n+1)L$ iterations, [BGT81].

With the perturbation of the original system and property (10), we do no longer require that P is bounded. As a byproduct, polyhedron P has not to be of full-dimension in the case that it is non empty; as system (11) is of full-dimension independent of whether P is or not, assuming that $P \neq \emptyset$. As a consequence, the basic ellipsoid algorithm can be generalized to apply for any polyhedron P and the two major assumptions are no longer necessary.

During all of the reasoning, we assumed to have exact arithmetic, meaning that no rounding errors during the computation are allowed. This implies that all data have to be stored in a mathematically correct way. As we use the Turing-machine concept for the running time analysis, we require that all computations have to be done in finite precision. Let us now have a closer look for the reason why this is crucial for ellipsoid algorithm 2.1.

The presentation of the ellipsoid with the matrix B_k in (3) yields to the convenient update formulas for the new ellipsoid, parameterized by B_{k+1} and x^{k+1}. However, to obtain the new center x^{k+1} one has to divide by factor $\sqrt{a_j^\top B_k a_j}$. If we work with finite precision, rounding errors are

the consequence, and it is likely that matrix B_k is no longer positive definite. This may cause that $a_j^\top B_k a_j$ becomes zero or negative, implying that the ellipsoid algorithm fails.

Hence, to implement the ellipsoid method, one has to use some modifications to make it numerically stable. One basic idea is to use factorization

$$B_k = L_k D_k L_k^\top$$

for the positive definite matrix B_k, with L being a lower triangular matrix with unit diagonal and diagonal matrix D_k with positive diagonal entries. Obtaining such a factorization is quite expensive as it is of order n^3. But there are update formulae applying for the case of the ellipsoid algorithm which have only quadratic complexity. Already in 1975, for such a type of factorization, numerically stable algorithms have been developed, insuring that D_k remains positive definite, see [GMS75]. We skip the technical details here and refer instead to GOLFARB and TODD, [GT82].

With a method at hand which can handle the ellipsoid algorithm in finite precision numerically stable, the proof of its polynomial running time is complete.

2.3 Modifications

In this subsection, we briefly discuss some straightforward modifications of the presented ellipsoid method in order to improve its convergence rate. Therefore, let us consider Figure 2 once more. From this figure it is intuitively clear that an ellipsoid containing set $\{x \in E_k \,|\, a_j^\top x \leq b\}$ has a smaller volume than the one containing the complete half-ellipsoid (7). In the survey paper of BLAND et al. it is shown that the smallest ellipsoid, arising from the so called deep cut $a_j^\top x \leq b$, can be obtained by choosing the following parameters for Algorithm 2.1

$$\begin{aligned} \tau &:= \frac{1+n\alpha}{n+1}, \\ \sigma &:= \frac{2+2n\alpha}{(n+1)(1+\alpha)}, \\ \delta &:= \frac{n^2}{(n^2-1)(1-\alpha^2)}, \end{aligned}$$

with

$$\alpha := \frac{a_j^\top - b_j}{\sqrt{a_j^\top B_k a_j}}.$$

The parameter α gives an additional advantage of this deep cut, as it is possible to check infeasibility or for redundant constraints, [Sch83].

Another idea could be to use a whole system of violated inequalities as a cut instead of only one. Such type of cuts are called surrogate cuts and were discussed by Goldfarb and Todd. An iterative procedure to generate these cuts was described by Krol and Mirman, [KM80].

Consider now the case that the inequality system (1) contains two parallel constraints which means that they differ only in the right hand side. With this it is possible to generate a new ellipsoid containing the information of both inequalities. These cuts are called *parallel cuts*. Update formulas for B_k and x^k were discovered independently by several authors. For more details, we refer to [Sch83, BGT81].

However, all modifications which have been found so far do not allow to reduce the worst case running time significantly – they especially do not allow to avoid the presence of L. This implies that the running time does not only depend on the size of the problem but also on the magnitude of the data.

At the end of the second chapter, we point out that the Ellipsoid Method can also be generalized to use other convex structures as ellipsoids. Methods working for instance only with spheres, or triangles, are possible. The only crucial point is that one has to make sure that its polynomial running time can be proven. Furthermore, the underlying polytope can be generalized to any convex set; for which the separation problem can be solved in polynomial time, see Section 3.2.

3 Applications

In 1981, Grötschel, Lovász and Schrijver used the Ellipsoid Method to solve many open problems in combinatorial optimization. They developed polynomial algorithms, for instance, for the vertex packing in perfect graphs, and could show that the weighted fractional chromatic number is $\mathcal{NP}$-*hard*, [GLS81]. Their proofs were mainly based on the relation of separation and optimization, which could be established with the help of the Ellipsoid Method. We discuss this topic in Section 3.2 and give one application for the maximum stable set problem. For all other interesting results, we refer to [GLS81]. But first, we consider another important application of the Ellipsoid Method. We examine two concepts showing the equivalence of solving a system of linear inequalities and to find an optimal solution to a LP. This will prove that LP is polynomial solvable.

3.1 Linear Programming

As we have seen in the last section, the Ellipsoid Method solves the problem of finding a feasible point of a system of linear inequalities. This problem is closely related to the problem of solving

the linear program

$$\begin{aligned} \max_{x} \quad & c^\top x \\ s.t. \quad & A^\top x \leq b, \\ & x \geq 0. \end{aligned} \tag{12}$$

Again, we assume that all data are integral. In the following we briefly discuss two methods of how the optimization problem (12) can be solved in polynomial time via the Ellipsoid Method. This will show that LP is in the class of polynomially solvable problems.

From duality theory, it is well known that solving the linear optimization problem (12) is equivalent to finding a feasible point of the following system of linear inequalities

$$\begin{aligned} A^\top x &\leq b, \\ -x &\leq 0, \\ -A\,y &\leq -c, \\ -y &\leq 0, \\ -c^\top x + b^\top y &\leq 0. \end{aligned} \tag{13}$$

The third and fourth inequality come from the dual problem of (12), insuring primal and dual feasibility of x and y, respectively. The last inequality results from the Strong Duality Theorem, implying that this inequality always has zero slack. The equivalence of the two problems means in this case that vector $\overline{x}$ of each solution pair $(\overline{x}, \overline{y})$ of (13) is an optimal solution of problem (12) and $\overline{y}$ is an optimum for the dual problem of (12). In addition, to each solution of the optimization problem exists a vector such that this pair is feasible for problem (13).

From the equivalence of the two problems (12) and (13) we immediately conclude that the linear programming problem can be solved in polynomial time; as the input data of (13) are polynomially bounded in the length of the input data of (12). This argument was used by Gács and Lovász in their accomplishment to Khachiyan's work, see [GL81]. The advantage of this method is that the primal and dual optimization problem are solved simultaneously. However, note that with this method, one has no idea whether the optimization problem is infeasible or unbounded in the case when the Ellipsoid Method proves that problem (13) is infeasible. Another disadvantage is that the dimension of the problem increases from n to $n + m$.

Next we discuss the so called bisection method which is also known as binary search or sliding objective hyperplane method. Starting with an upper and lower bound of an optimal solution, the basic idea is to make the difference between the bounds smaller until they are zero or small enough. Solving system $A^\top x \leq b$, $x \geq 0$ with the Ellipsoid Method gives us either a vector $\overline{x}$ providing the lower bound $l := c^\top \overline{x}$ for problem (12) or in the case that the polyhedron is empty, we know that

the optimization problem is infeasible. An upper bound can be obtained, for instance, by finding a feasible vector to the dual problem $A\,y \geq c$, $y \geq 0$. If the Ellipsoid Method proves that the polytope of the dual problem is empty, we can use the duality theory (as we already know that problem (12) is not infeasible) to conclude that the optimization problem (12) is unbounded. In the other case we obtain vector $\overline{y}$ yielding to the upper bound $u := b^\top \overline{y}$ of problem (12), according to the Weak Duality Theorem. Once bounds are obtained, one can iteratively use the Ellipsoid Method to solve the modified problem $A^\top x \leq b$, $x \geq 0$ with the additional constraint

$$-c^\top x \leq -\frac{u+l}{2},$$

which is a constraint on the objective function value of the optimization problem. If the new problem is infeasible, one can update the upper bound to $\frac{u+l}{2}$, and in the case that the ellipsoid algorithm computes a vector $\overline{x}$, the lower bound can be increased to $c^\top \overline{x}$ which is greater or equal to $\frac{u+l}{2}$. In doing so, one at least bisects the gap in each step. However, this method does not immediately provide a dual solution. Note that only one inequality is added during the process, keeping the problem size small. More details and especially the polynomial running time of this method are discussed by Padberg and Rao, [aMRR80].

3.2 Separation and Optimization

An interesting property of the ellipsoid algorithm is that it does not require an explicit list of all inequalities. In fact, it is enough to have a routine which solves the so called Separation Problem for a convex body K:

Given $z \in \mathbb{R}^n$, either conclude that $z \in K$ or give a vector $\pi \in \mathbb{R}^n$ such that inequality $\pi^\top x < \pi^\top z$ holds for all $x \in K$.

In the latter case we say that vector π separates z from K. In the following, we restrict the discussion to the case when K is a polytope meeting the two assumptions of Section 2.1. To be consistent with the notation, we write P for K. From the basic ellipsoid algorithm 2.1 follows immediately that if one can solve the separation problem for polytope P polynomially in L and n, then the corresponding optimization problem

$$\begin{aligned} \max_x \quad & c^\top x \\ s.t. \quad & x \in P \end{aligned}$$

can also be solved in polynomial time in L and n; L is again the encoding length of the input data, see Section 2.1. The converse statement was proven by Groetschel et al., yielding to the following equivalence of separation and optimization:

The separation problem and the optimization problem over the same family of polytopes are polynomially equivalent.

Consider now an example to see how powerful the concept of separation is. Given a graph $G = (V, E)$ with node set V and edges $e \in E$. A stable set S of graph G is defined as a subset of V with the property that any two nodes of S are not adjacent; which means that no edge between them exists in E. To look for a maximum one is the maximum stable set problem. This is a well known optimization problem and proven to be $\mathcal{NP}$-*hard*, see [GJ79]. It can be modeled, for instance, as the integer program

$$\begin{aligned} \max_{x} \quad & c^\top x \\ s.t. \quad & x_i + x_j \leq 1 \qquad \forall (i,j) \in E \\ & x \in \{0,1\} \end{aligned} \tag{14}$$

with incidence vector x, meaning that $x_i = 1$ iff node i is in a maximum stable set, otherwise it is zero. Constraints (14) are called edge inequalities. Relaxing the binary constraints for x gives

$$0 \leq x \leq 1 \tag{15}$$

yielding to a linear program. However, this relaxation is very weak; consider therefore a complete graph. To improve it, one can consider the odd-cycle inequalities

$$\sum_{i \in C} x_i \leq \frac{|C| - 1}{2} \tag{16}$$

for each odd cycle C in G. Recognize that there are in general exponentially many such inequalities in the size of graph G. Obviously, every stable set satisfies them and hence they are valid inequalities for the stable set polytope. The polytope satisfying the trivial-, edge- and odd-cycle inequalities

$$P := \{x \in \mathbb{R}^{|V|} \,|\, x \text{ satisfies } (14), (15) \text{ and } (16)\} \tag{17}$$

is called the cycle-constraint stable set polytope. Notice that this polytope is contained strictly in the stable set polytope. It can be shown that the separation problem for polytope (17) can be solved in polynomial time. One idea is based on a construction of an auxiliary graph H with a double number of nodes. Solving a sequence of n shortest path problems on H solves then the separation problem with a total running time of order $|V| \cdot |E| \cdot \log(|V|)$. With the equivalence of optimization and separation, the stable set problem over the cycle-constraint stable set polytope can be solved in polynomial time. This is quite a remarkable conclusion as the number of odd-cycle inequalities may be exponential. However, note that it does not imply that the solution will be integral and hence, we cannot conclude that the stable set problem can be solved in polynomial time. But we can conclude that the stable set problem for t-perfect graphs[1] can be solved in polynomial time. For more details about this topic see, for instance, [GS86, Reb06].

4 Conclusion

In 1980, the Ellipsoid Method seemed to be a promising algorithm to solve problems practically [Tee80]. However, even though many modifications to the basic ellipsoid algorithm have been made,

[1] A graph is called t-perfect, iff the stable set polytope is equal to the cycle-constraint stable set polytope (17)

the worst case running time still remains a function in n, m and especially L. This raises two main questions. First, is it possible to modify the ellipsoid algorithm to have a running time which is independent of the magnitude of the data, but instead depends only on n and m – or at least any other algorithm with this property solving LPs? (This concept is known as strongly polynomial running time.) The answer to this question is still not known and it remains an open problem. In 1984, KARMARKAR introduced another polynomial running time algorithm for LP which was the start of the Interior Point Methods, [Kar84]. But also his ideas could not be used to solve this question. For more details about this topic see [NW99, Tar86]. The second question, coming into mind, is how the algorithm performs in practical problems. Unfortunately, it turns out that the ellipsoid algorithm tends to have a running time close to its worst-case bound and is inefficient compared to other methods. The Simplex Method developed by GEORGE B. DANTZIG in 1947, was proven by KLEE and MINTY to have exponential running time in the worst case, [KM72]. In contrast, its practical performance is much better and it normally requires only a linear number of iterations in the number of constraints.

Until now, the Ellipsoid Method has not played a role for solving linear programming problems in practice. However, the property that the inequalities themselves have not to be explicitly known, distinguishes the Ellipsoid Method from others, for instance from the Simplex Method and the Interior Point Methods. This makes it a theoretically powerful tool, for instance attacking various combinatorial optimization problems which was impressively shown by GRÖTSCHEL, LOVÁSZ and SCHRIJVER.

5 Crossreferences

Linear programming, LP; Linear programming: Interior point methods; Linear programming: Karmakar projective algorithm; Linear programming: Klee-Minty examples; Polynomial optimization; Volume computation for polytopes: Strategies and performances

References

[aMRR80] M. W. Padberg and M. R. Rao. The Russian Method for Linear Inequalities. Graduate School of Business Administration, New York University, January 1980.

[BGT81] Robert G. Bland, Donald Goldfarb, and Michael J. Todd. The Ellipsoid Method: A Survey. *Operations Research*, 29(6):1039–1091, 1981.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability, A guide to the Theory of NP-Completeness*. A series of books in the mathematical sciences, Vicot Klee, Editor. W. H. Freeman and Company, 1979.

[GL81] P. Gács and L. Lovász. Khachiyan's Algorithm for Linear Programming. *Mathematical Programming Study*, 14:61–68, 1981.

[GLS81] Martin Grötschel, László Lovász, and Alexander Schrijver. The Ellipsoid Method and Its Consequences in Combinatorial Optimization. *Combinatorica*, 1:169–197, 1981.

[GLS88] Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Algorithms and Combinatorics 2. Springer, 1988.

[GMS75] Philip E. Gill, Walter Murray, and Michael A. Saunders. Methods for Computing and Modifying the LDV Factors of a Matrix. *Mathematics of Computation*, 29(132):1051–1077, October 1975.

[GS86] A. M. H. Gerards and A. Schrijver. Matrices with the Edmonds-Johnson property. *Combinatorica*, 6(4):365–379, 1986.

[GT82] Donald Goldfarb and Michael J. Todd. Modifications and Implementation of The Ellipsoid Algorithm for Linear Programming. *Mathematical Programming*, 23:1–19, 1982.

[Kar84] N. Karmarkar. A New Polynomial-Time Algorithm for Linear Programming. *Combinatorica*, 4(4):373–395, 1984.

[Kha79] Leonid G. Khachiyan. A Polynomial Algorithm in Linear Programming. *Doklady Akademiia Nauk SSSR*, 244:1093–1096, 1979. (English translation: Soviet Mathematics Doklady, 20(1):191–194, 1979).

[Kha80] Leonid G. Khachiyan. Polynomial Algorithms in Linear Programming. *Zhurnal Vychislitel'noi Mathematiki i Mathematicheskoi Fiziki*, 20:51–68, 1980. (English translation: USSR Computational Mathematics and Mathematical Physics, 20(1):53–72, 1980).

[KM72] V. Klee and G. L. Minty. How good is the Simplex Algorithm? *Inequalities III*, pages 159–175, 1972. O. Shisha (editor). Academic Press, New York.

[KM80] J. Krol and B. Mirman. Some Practical Modifications of The Ellipsoid Method for LP Problems. Arcon Inc, Boston, Mass, USA, 1980.

[Law80] Eugene L. Lawler. The Great Mathematical Sputnik of 1979. *Sciences*, 20(7):12–15, 34–35, September 1980.

[MT05] Walter Murray and Michael Todd. Tributes to George Dantzig and Leonid Khachiyan. *SIAM Activity Group on Optimization - Views and News*, 16(1–2):1–6, October 2005.

[NW99] George L. Nemhauser and Laurence A. Wolsey. *Integer and Combinatorial Optimization*. Wiley-Interscience Series in Discrete Mathematics and Optimization. Johnson Wiley & Sons, Inc., 1999.

[Reb06] Steffen Rebennack. Maximum Stable Set Problem: A Branch & Cut Solver. Diplomarbeit, Ruprecht–Karls Universität Heidelberg, Heidelberg, Germany, 2006.

[Sch83] R. Schrader. The Ellipsoid Method and Its Implications. *OR Spektrum*, 5(1):1–13, March 1983.

[Sho77] N. Z. Shor. Cut-off Method with Space Extension in Convex Programming Problems. *Kibernetika*, 13(1):94–95, 1977. (English translation: Cybernetics, 13(1):94–96, 1977).

[Sia05] Leonid Khachiyan, 1952-2005: An Appreciation. *SIAM News*, 38(10), December 2005.

[Tar86] Eva Tardos. A Strongly Polynomial Algorithm to Solve Combinatorial Linear Programs. *Operations Research*, 34(2):250–256, 1986.

[Tee80] G. J. Tee. Khachian's efficient algorithm for linear inequalities and linear programming. *SIGNUM Newsl.*, 15(1):13–15, 1980.

[YN76a] D. B. Yudin and A. S. Nemirovskiĭ. Evaluation of the Informational Complexity of Mathematical Programming Problems. *Èkonomika i Mathematicheskie Metody*, 12:128–142, 1976. (English translation: Matekon, 13(2):3–25, 1976–7).

[YN76b] D. B. Yudin and A. S. Nemirovskiĭ. Informational Complexity and Efficient Methods for the Solution of Convex Extremal Problems. *Èkonomika i Mathematicheskie Metody*, 12:357–369, 1976. (English translation: Matekon, 13(3):25–45, 1977).

[YN77] D. B. Yudin and A. S. Nemirovskiĭ. Optimization Methods Adapting to the "Significant" Dimension of the Problem. *Automatika i Telemekhanika*, 38(4):75–87, 1977. (English translation: Automation and Remote Control, 38(4):513–524, 1977).