THE ELLIPSOID METHOD: A SURVEY+

Robert G. Bland*
Donald Goldfarb**
Michael J. Todd***

TR 80-441
August 1980

Department of Computer Science
Cornell University
Ithaca, NY 14853

+This report is also available as Technical Report No. 476, School of Operations Research and Industrial Engineering, College of Engineering, Cornell University, Ithaca, New York, 14853

*Sloan Foundation Research Fellow. Partially supported by the National Science Foundation under grant ENG-7910807.

**Partially supported by the U.S. Army Research Office under grant DAAG29-77-G-0114 and the National Science Foundation under grand MCS-8006065.

***Partially supported by the National Science Foundation under grant ECS-7921279.

Abstract

In February 1979 a note by L.G. Khachiyan indicated how an ellipsoid method for linear programming can be implemented in polynomial time. This result has caused great excitement and stimulated a flood of technical papers. Ordinarily there would be no need for a survey of work so recent. The current circumstances are obviously exceptional. Word of Khachiyan's result has spread extraordinarily fast, much faster than comprehension of its significance. A variety of issues have in general not been well understood, including the exact character of the ellipsoid method and of Khachiyan's result on polynomiality, its practical significance in linear programming, its implementation, its potential applicability to problems outside of the domain of linear programming, and its relationship to earlier work. Our aim here is to help clarify these important issues in the context of a survey of the ellipsoid method, its historical antecedents, recent developments, and current research.

1. Introduction

In February 1979 the note "A polynomial algorithm in linear programming" by L.G. Khachiyan [35] appeared in Doklady Akademiia Nauk SSSR. Several months later it first came to the attention of operations researchers, computer scientists, and mathematicians in the West through informal dissemination and discussion. In another six months Khachiyan's note became front-page news, not only for researchers, but for readers of major daily newspapers in the United States, Europe, and Japan (see [64]).

The Theoretical Result

The immediate significance of Khachiyan's article was the resolution of an important theoretical question concerning the computational complexity of linear programming. Most of the basic discrete optimization problems in operations research have been known for a number of years either to be solvable in polynomial-time (e.g., the shortest path problem with nonnegative arc lengths), or to be *NP*-complete (e.g., the traveling salesman problem, and the shortest path problem with arbitrary arc lengths).* Yet linear programming, the most studied of all optimization problems in operations research, resisted classification. Most researchers considered it very unlikely that linear programming might be theoretically as difficult as the *NP*-complete problems, but no one had managed to prove its membership in *P*, the class of problems solvable by polynomial-time algorithms. Finally, Khachiyan [35] indicated how one could adapt a method for convex optimization developed by the Soviet mathematicians N.Z. Shor, D.B. Iudin, and A.S. Nemirovskii to devise a polynomial-time ellipsoid method for linear programming.

---

*Appendix A provides an informal discussion of the notions of polynomial boundedness and *NP*-completeness for the unacquainted reader. For a rigorous treatment see [3,15,33,34].

of n^2. Since such a graph may have as many as $\frac{1}{2}(n^2-n)$ arcs, Dijkstra's algorithm appears to be an attractive practical procedure. An examination of the known polynomial bound on the ellipsoid method for linear programming does not lead so readily to a promising conclusion, as we shall see.

We must also keep in mind that polynomial-boundedness is a worst-case criterion; the most perverse problem instances determine this measure of an algorithm's performance. How likely are we to encounter in practice problem instances like those in $\{Q_n\}$ that cause algorithm A to behave badly? Are they pathological, contrived? This has been claimed of those known families of problems that lead to exponential behavior of the standard simplex pivoting rules.

Researchers in computational complexity are very well aware of these limitations to the practical significance of polynomiality. However, most known polynomial-time algorithms for problems of interest to operations researchers are, in fact, efficient in practice as well as in theory, perhaps leading some to attach greater significance to polynomiality than is merited.

Outline

Ordinarily there would be no need for a survey of work so recent as that prompted by [35]. The current circumstances are obviously exceptional. Word of Khachiyan's result has spread extraordinarily fast, much faster than comprehension of its significance. A variety of issues have been so muddled by accounts in the press, that even a technically sophisticated reader may be uncertain of the exact character of the ellipsoid method and of Khachiyan's result on polynomiality, its practical significance in linear programming, its implementation, its potential applicability to problems outside of the domain of linear programming, and its relationship to earlier work. Our aim here is to help clarify these important issues in the context of a survey of the ellipsoid method, its historical antecedents, recent developments, and current research.

In Section 2 we describe the basic ellipsoid algorithm for finding a feasible solution to a system of linear inequalities. We outline the modifications introduced by Khachiyan and the arguments used by him to prove that the feasibility or infeasibility of such a system can be determined in polynomial time with this algorithm. The extension to linear optimization is discussed in Section 5.

In Section 3 we present a detailed account of the research that led up to the ellipsoid algorithm. We show that it was a fairly natural outgrowth of the relaxation and subgradient algorithms of Agmon, Motzkin and Schoenberg, and Shor, the method of centered cross-sections of Levin, and the methods of space dilation of Shor. In particular, we observe that the ellipsoid algorithm was first introduced by the Soviet mathematicians D.B. Iudin and A.S. Nemirovskii and then clarified by N.Z. Shor; all three were interested in its application to convex, not necessarily differentiable, optimization. Khachiyan modified the method to obtain a polynomial-time algorithm for the feasibility problem for a system of linear inequalities.

If the ellipsoid algorithm is to be more than just a theoretical tool, it must be implemented in a numerically stable way and modified so as to increase its rate of convergence. In Section 4, three modifications to the basic algorithm are described. These are the use of deep

$$A^T x \leq b \tag{2.1}$$

where A^T is $m \times n$ and b is an m-vector. The columns of A, corresponding to outward normals to the constraints, are denoted $a_1, a_2, \ldots, a_m$, and the components of b are denoted $\beta_1, \beta_2, \ldots, \beta_m$. Thus (2.1) can be restated as

$$a_i^T x \leq \beta_i, \quad i = 1, 2, \ldots, m.$$

We assume throughout that n is greater than one.

<u>The Basic Iteration</u>

The ellipsoid method constructs a sequence of ellipsoids $E_0, E_1, \ldots, E_k$, $\ldots$, each of which contains a point satisfying (2.1), if one exists. On the (k+1)st iteration, the method checks whether the center x_k of the current ellipsoid E_k satisfies the constraints (2.1). If so, the method stops. If not, some constraint violated by x_k, say

$$a^T x \leq \beta \tag{2.2}$$

is chosen and the ellipsoid of minimum volume that contains the half-ellipsoid

$$\{x \in E_k \mid a^T x \leq a^T x_k\} \tag{2.3}$$

is constructed. (See Figure 1(a).) This new ellipsoid and its center are denoted by E_{k+1} and x_{k+1}, respectively, and the above iterative step is repeated.

Except for initialization, this gives a (possibly infinite) iterative algorithm for determining the feasibility of (2.1). Khachiyan showed that one can determine whether (2.1) is feasible or not within a prespecified number of iterations by: (i) modifying this algorithm to account for finite precision arithmetic; (ii) applying it to a suitable perturbation of system (2.1); and (iii) choosing E_0 appropriately. System (2.1) is feasible if and only if termination occurs with a feasible solution of the perturbed system within the prescribed number of iterations.

Algebraically, we can represent the ellipsoid E_k as

$$E_k = \{x \in R^n \mid (x-x_k)^T B_k^{-1} (x-x_k) \leq 1\} \tag{2.4}$$

where x_k is its center and B_k is a positive definite symmetric matrix. In terms of this representation the (k+1)st iterative step of the ellipsoid method is simply given by the formulae

$$x_{k+1} = x_k - \tau \frac{B_k a}{\sqrt{a^T B_k a}} \tag{2.5}$$

and

$$B_{k+1} = \delta\left(B_k - \sigma \frac{B_k a (B_k a)^T}{a^T B_k a}\right) \tag{2.6}$$

where

$$\tau = 1/(n+1), \quad \sigma = 2/(n+1), \quad \text{and} \quad \delta = n^2/(n^2-1). \tag{2.7}$$

That E_{k+1} determined by x_{k+1} and B_{k+1} as in (2.4)-(2.7) is the ellipsoid of smallest volume that contains the half-ellipsoid (2.3) is proved in Appendix B. We call τ, σ, and δ the step, dilation, and expansion parameters, respectively. Note that if B_k is a multiple of

We need to show that the number of steps is polynomial in L. There are two main issues in the proof.

First, the formulae (2.5)-(2.7) for x_{k+1} and B_{k+1} assume exact arithmetic. To perform an algorithm in polynomial time with accepted models of computation one must use finite-precision arithmetic. Khachiyan indicated that $23L$ bits of precision before the point and $38nL$ after suffice. Note that if the values of x_{k+1} and B_{k+1} are rounded to this specified number of bits, the ellipsoid E_{k+1} may not contain the required half-ellipsoid. Khachiyan showed that if B_{k+1} is multiplied by a factor slightly larger than one before being rounded, then E_{k+1} will still contain this half-ellipsoid. Unless otherwise noted we will assume throughout that exact arithmetic is used.

Second, we must provide a polynomial bound on the number of iterations required. We start by examining a special case in which for some known $a_0 \in \mathbb{R}^n$ and $R > r > 0$, and unknown $a^* \in \mathbb{R}^n$.

$$S(a^*,r) \subseteq P \subseteq S(a_0,R), \tag{2.10}$$

where P is the solution set of (2.1) and $S(\alpha,\rho)$ denotes the ball of radius ρ centered at α. In this case we initialize with $E_0 = S(a_0,R)$. We now use the fact that when the formulae (2.4)-(2.7) are employed,

$$\frac{\text{vol } E_{k+1}}{\text{vol } E_k} = \frac{n}{n+1}\left(\frac{n^2}{n^2-1}\right)^{\frac{n-1}{2}} < e^{-1/2(n+1)}. \tag{2.11}$$

Suppose $k > 2n(n+1)\log(R/r)$. Then, assuming the ellipsoid algorithm continues this far, the volume of E_k will have shrunk to less than $(r/R)^n$

times its original value; thus it cannot contain a ball of radius r. But, as we have remarked above, the sequence of ellipsoids generated satisfies $S(a^*,r) \subseteq P \subseteq E_k$ for all k. This contradiction proves that the algorithm must terminate with $x_k \in P$ for some $k \leq 2n(n+1)\log(R/r)$. Hence if R and r in (2.10) are regarded as part of the input, or $\log(R/r)$ is polynomial in L, then the number of steps required is polynomial.

In many practical problems, a priori bounds as in (2.10) may be available and should be used. However, to provide a polynomial algorithm we must assume that nothing is known about the system (2.1) other than its description in terms of A and b. In this case Khachiyan proved that, if P is nonempty, then

$$P \cap S(0,2^L) \neq \emptyset; \qquad (2.11)$$

thus 2^L can play the role of R in (2.10), and we can initialize the algorithm with $E_0 = S(0,2^L)$. Clearly, however, P need not contain a ball of any positive radius. Thus Khachiyan perturbed (2.1) to obtain the system of inequalities

$$2^L a_i^T x \leq 2^L \beta_i + 1, \quad i = 1,2,\ldots,m \qquad (2.12)$$

with solution set P'. He then proved that P is nonempty if and only if P' is. Moreover, a solution to (2.1) can be obtained (in polynomial time) from a solution to (2.12)* and the number of bits needed to represent (2.12) is at most $(m(n+1)+1)L$, hence polynomial in L. The reason for considering

---

*One method for obtaining exact solutions from approximate solutions is described at the end of Section 5.

The subgradient method for minimizing a convex, not necessarily differentiable, function $f: \mathbb{R}^n \to \mathbb{R}$ was apparently first introduced by Shor [51]. It has the general form

$$x_{k+1} = x_k - \mu_k g_k / ||g_k|| \tag{3.2}$$

where g_k is a subgradient of the function f at x_k. Note that if we wish to solve (2.1) we can minimize

$$f(x) = \max_i \{a_i^T x - \beta_i, 0\}; \tag{3.3}$$

then $a = a_i$ is a subgradient of f at x_k if $a_i^T x \leq \beta_i$ is a most-violated constraint from (2.1). Thus (3.2) includes (3.1) as a special case. Ermolev [12] and Polyak [49] give choices for μ_k that ensure global convergence; for example, $\mu_k \to 0$ and $\Sigma \mu_k = \infty$ suffice. However very slow convergence results. Polyak [50] and Shor [52] demonstrate linear convergence for certain choices of the step lengths μ_k under suitable conditions on f. However, the rate of convergence is still heavily dependent on the function f.

Shor [53,54] seems to have been the first to realize that improvements could be made by working in a transformed space. The idea is exactly that which leads from the steepest descent algorithm (with linear convergence, the rate depending on the function) to Newton's method (with quadratic convergence for smooth functions) and quasi-Newton algorithms (with superlinear convergence for smooth functions). The iteration now takes the form

$$\tilde{g}_k = J_k^T g_k / ||J_k^T g_k||,$$

$$x_{k+1} = x_k - \alpha_k J_k \tilde{g}_k, \qquad (3.4)$$

$$J_{k+1} = J_k(I - \beta_k \tilde{g}_k \tilde{g}_k^T),$$

for suitable parameters α_k, β_k. The update of the matrix J_k corresponds to "space dilation" in the direction of the subgradient g_k. Shor [54] describes precisely the difficulties with the linear convergence rate of his earlier subgradient method [52]. His modified algorithm (3.4), when f satisfies certain conditions allowing the parameters α_k and β_k to be estimated, provides linear convergence whose rate depends on the function f, but is invariant with respect to linear transformations. When f is quadratic and strictly convex, the parameters can be chosen so that the method becomes a method of conjugate gradients [53]. For this algorithm, the minimum value of f must be known; his later method [54] relaxes this requirement. In [58], Shor and Zhurbenko perform the "space dilation" in the direction of the difference $y_k = g_{k+1} - g_k$ between successive subgradients; this method is even more reminiscent of quasi-Newton minimization methods. This paper contains results of some limited computational experiments.

The last method on which the ellipsoid algorithm is based is that of Levin [43] who addressed the problem of minimizing a convex function f over a bounded polyhedron $P_0 \subseteq R^n$. The method produces a sequence of iterates $\{x_k\}$ and polytopes $\{P_k\}$ by choosing x_k as the center of gravity of P_k and

Khachiyan's contributions [35] were precisely those described in Section 2. He gave a modified form of the ellipsoid algorithm for the feasibility problem of (2.1) with integer data and showed that feasibility or infeasibility could be determined in polynomial time by this algorithm.

We conclude this section by mentioning other related papers in the Soviet literature. Shor has two survey papers [55,57] on non-differentiable optimization. The second of these [57] states that computational comparisons of subgradient algorithms, subgradient algorithms with space dilation in the direction of the subgradient [53,54] or in the direction of the difference of successive subgradients [58], and the "conjugate subgradient" methods of Lemarechal [42] and Wolfe [63] tend to favor the method [58], at least for dimensions up to 200-300. For higher dimensions, storing and updating the extra space dilation matrix becomes too expensive and subgradient or conjugate subgradient methods become preferable. The paper also demonstrates the application of the ellipsoid algorithm to computing saddle points.

The interesting paper [28] by Nemirovskii and Iudin concerns an application of the ellipsoid method where the "effective" dimension may be much less than n. In this case a projection method can lead to faster convergence. Finally we note that, for Shor's earlier method [53,54], Skokov [59] suggested updating the symmetric matrix $B_k = J_k J_k^T$ to save storage and reduce computation. His formulae are analogous to those of Gács and Lovasz [14] which we have given as (2.5)-(2.7). We will show in Section 6 some of the dangers of this approach.

4. Modifications of the Basic Algorithm

In this section we describe several simple modifications to the basic algorithm to improve its rate of convergence.

Deep Cuts

As before, suppose that x_k violates constraint (2.2). The ellipsoid E_{k+1} determined by formulae (2.5)-(2.7) contains the half-ellipsoid $\{x \in E_k | a^T x \leq a^T x_k\}$. As we only require that E_{k+1} contain the smaller portion of E_k, $\{x \in E_k | a^T x \leq \beta\}$, it seems obvious that we can obtain an ellipsoid of smaller volume by using the "deep cut" $a^T x \leq \beta$ instead of the cut $a^T x \leq a^T x_k$, which passes through the center of E_k. This is illustrated in Figures 1(a) and 1(b). The smallest such ellipsoid is given by x_{k+1} and B_{k+1} as in (2.5)-(2.6) with the parameters τ, σ, and δ chosen as

$$\tau = \frac{1+n\alpha}{n+1}, \quad \sigma = \frac{2(1+n\alpha)}{(n+1)(1+\alpha)}, \quad \text{and} \quad \delta = \frac{n^2}{n^2-1}(1-\alpha^2) \qquad (4.1)$$

where

$$\alpha = \frac{a^T x_k - \beta}{\sqrt{a^T B_k a}}. \qquad (4.2)$$

For a proof of this see Appendix B.

The quantity α which now appears in the updating formulae represents the (algebraic) distance of x_k from the half-space $H = \{x \in \mathbb{R}^n | a^T x \leq \beta\}$ in the metric corresponding to the matrix B (i.e., $||y||_B = (y^T B^{-1} y)^{1/2}$). Another way of viewing α is to represent the ellipsoid E_k as

$$E_k = \{x \in \mathbb{R}^n | x = x_k + J_k z, \ ||z|| \leq 1\} \qquad (4.3)$$

respect to that subset. It is shown in [21], [61] that if $\bar{A}^T B_k \bar{A}$ has nonpositive off-diagonal entries--i.e., the constraint normals in $\bar{A}$ are mutually obtuse in the metric given by B_k--and if x_k violates all constraints in (4.5), then the $\bar{u}$ given by (4.6) is nonnegative.

Solving a quadratic programming problem or computing $\bar{u}$ by (4.6) for a large subset of constraints may be too high a price to pay to obtain the deepest or nearly deepest surrogate cut. Consequently in [21] it is recommended that only surrogate cuts which can be generated from two constraints be considered. In [38], surrogate cuts are generated from as many as n constraints by an iterative procedure which combines two constraints at a time.

Parallel Cuts

If the system of linear inequalities (2.1) contains a parallel pair of constraints

$$a^T x \leq \beta \quad \text{and} \quad -a^T x \leq -\hat{\beta}$$

it is possible to use both constraints simultaneously to generate the new ellipsoid E_{k+1}. Let $\alpha = \frac{a^T x_k - \beta}{\sqrt{a^T B_k a}}$ and $\hat{\alpha} = \frac{\hat{\beta} - a^T x_k}{\sqrt{a^T B_k a}}$, and suppose that $\alpha\hat{\alpha} < 1/n$ and $\alpha \leq -\hat{\alpha} \leq 1$. Then formulae (2.5)-(2.6) with

$$\sigma = \frac{1}{n+1}\left(n + \frac{2}{(\alpha-\hat{\alpha})^2}\left(1 - \alpha\hat{\alpha} - \rho/2\right)\right),$$

$$\tau = \left(\frac{\alpha-\hat{\alpha}}{2}\right)\sigma, \qquad \delta = \frac{n^2}{n^2-1}\left(1 - \frac{\alpha^2 + \hat{\alpha}^2 - \rho/n}{2}\right),$$

$$\text{and} \quad \rho = \sqrt{4(1-\alpha^2)(1-\hat{\alpha}^2) + n^2(\hat{\alpha}^2-\alpha^2)^2}$$

generate an ellipsoid that contains the slice $\{x \in E_k | \hat{\beta} \leq a^T x \leq \beta\}$ of E_k. When $\hat{\beta} = \beta$, i.e., $a^T x = \beta$ for all feasible x, $\hat{\alpha} = -\alpha$ and we get $\tau = \alpha$, $\sigma = 1$, and $\delta = \frac{n^2}{n^2-1}(1-\alpha^2)$; that is, $\text{rank}(B_{k+1}) = \text{rank}(B_k) - 1$ and E_{k+1} becomes flat in the direction of a.

5. Solving Linear Programming Problems

So far we have considered only the feasibility problem for systems of linear inequalities. Here we address the linear programming problem, which we write in inequality form as

$$\text{maximize } c^T x \text{ subject to } A^T x \leq b, \quad x \geq 0, \tag{5.1}$$

where A^T is $m \times n$. There are several ways this problem can be attacked by the ellipsoid algorithm. We first discuss these approaches from a theoretical viewpoint to establish the existence of polynomial algorithms for (5.1). Then we mention more practical considerations.

Simultaneous Solution of the Primal and Dual in $\mathbb{R}^{m+n}$

The problem dual to (5.1) is

$$\text{minimize } b^T y \text{ subject to } Ay \geq c, \quad y \geq 0. \tag{5.2}$$

By strong duality, (5.1) has a finite optimal solution if and only if (5.2) does, in which case the objective function values are equal. For any primal feasible x and dual feasible y we have $c^T x \leq b^T y$ by weak duality. Thus x^* and y^* are optimal solutions to (5.1) and (5.2)

It can be shown from (4.4) that for a given α, $r(\alpha)$, the ratio of the volumes of E_{k+1} and E_k, increases with the dimension n. The volume reduction from a cut based on the primal constraints will therefore be smaller in the product space $\mathbb{R}^{n+m}$ than in the primal space $\mathbb{R}^n$. Hence it is desirable to handle the objective function of (5.1) without increasing the dimension of the problem. We now discuss two such approaches based upon systems of linear inequalities of the form

$$A^T x \leq b, \quad -x \leq 0, \quad -c^T x \leq -\zeta \tag{5.4}$$

for various values of ζ. These methods do not produce optimal dual solutions.

<u>Bisection Method</u>

This method initially applies the ellipsoid algorithm to the constraints of (5.1) to obtain a feasible solution $\underline{x}$ if one exists; if there is none, we terminate. Then $\underline{\zeta} = c^T\underline{x}$ is a lower bound on the optimal value of (5.1). Next we obtain an upper bound on this value. If the feasible region of (5.1) is bounded and contained in the known ellipsoid E_0 given by (2.4) then $\bar{\zeta} = c^T x_0 + (c^T B_0 c)^{1/2}$ is such an upper bound. Otherwise, we may apply the ellipsoid algorithm to the constraints of (5.2) to obtain a dual feasible solution $\bar{y}$ if one exists, and set $\bar{\zeta} = b^T\bar{y}$; if (5.2) is infeasible, we terminate. From now on, each major iteration starts with an interval $[\underline{\zeta}, \bar{\zeta}]$ that contains the optimal value, where $\underline{\zeta} = c^T\underline{x}$ for some known feasible solution $\underline{x}$, and applies the ellipsoid algorithm to (5.4) with

$\zeta = (\bar{\zeta}+\underline{\zeta})/2$. If a feasible solution x_k is generated, we set $\underline{x} \leftarrow x_k$, $\underline{\zeta} \leftarrow c^T x_k$ and proceed to the next major iteration. If it is determined that (5.4) is infeasible, we set $\bar{\zeta} \leftarrow \zeta$ and proceed to the next iteration. The process stops when $\bar{\zeta} - \underline{\zeta}$ is sufficiently small. This algorithm can be made polynomial. It has the advantage of operating only in $\mathbb{R}^n$ (except for possibly one application in $\mathbb{R}^m$).

From a practical viewpoint, the main disadvantage is that the systems (5.4) with ζ too large will be infeasible and may take a large number of iterations. It is therefore imperative to use the deep cuts of Section 4 and the resulting tests for infeasibility to allow early termination in such cases. Note that when a major iteration is started with a new ζ greater than the old (i.e., a feasible solution $\underline{x}$ has just been generated), the final ellipsoid of the previous major iteration with center $\underline{x}$ can be taken as the initial ellipsoid of the new major iteration. If, instead ζ has just been decreased, we can initialize with the last recorded ellipsoid with a feasible center--the algorithm backtracks. Avoiding such backtracking leads to the final method that we shall consider.

Sliding Objective Function Method

Suppose we start as before by generating a feasible solution $\underline{x}$ to (5.1). We next consider (5.4) with $\zeta \leftarrow \underline{\zeta} = c^T\underline{x}$. While $\underline{x}$ is feasible in this problem, we may proceed with the ellipsoid algorithm since as long as the center is either on or violates the chosen cut, the next ellipsoid can be defined. Hence in this method, we are always considering feasible systems (except possibly the first). Whenever a feasible iterate x_k satisfies $c^T x_k > c^T\underline{x} = \underline{\zeta}$, we set $\underline{x} \leftarrow x_k$ and $\zeta \leftarrow \underline{\zeta} = c^T x_k$ and proceed.

then x^* is the unique rational vector of form (5.5) in this ball. For $y \in S(x, \frac{1}{2\Delta^2})$ implies that $||y-x^*|| < \frac{1}{\Delta^2}$, but if y is also of the form (5.5) and for some j, $y_j \neq x_j^*$, then $|y_j-x_j^*| \geq \frac{1}{\Delta^2}$ implying that $||y-x^*|| \geq \frac{1}{\Delta^2}$. Therefore if x, x^* are as above, x is known, and x^* is unknown, we can obtain x^* by rounding each component of x to the nearest rational $\frac{p}{q}$ having $|q| \leq \Delta$ by the method of continued fractions (see [46]).

We will be interested in the situation where x^* is an optimal extreme point of the linear programming problem (5.1), and x is obtained from the ellipsoid method. Grötschel, Lovász, and Schrijver [23] point out that one can replace the objective function vector c of (5.1) by a perturbed vector $d = \gamma^n c + (\gamma^0, \ldots, \gamma^{n-1})^T$ such that the problem

$$\text{maximize } d^T x \text{ subject to } Ax \leq b,\ x \geq 0 \tag{5.6}$$

has a unique optimal solution at an extreme point x^*, and x^* also solves (5.1); for example we can set $\gamma = 2^{Ln+L+1}$. It is important to note that $\log \gamma$ is polynomial in L and n, so that the size of (5.6) is a polynomial function of the size of (5.1). By Cramer's Rule x^* is of the form (5.5) for Δ greater than or equal to the absolute value of the largest determinant of any $n \times n$ submatrix of the constraint matrix of (5.1); in particular we can take $\Delta = 2^L$. Now we would like to be able to guarantee that for sufficiently small $\varepsilon > 0$

$$||x^*-x|| < \frac{1}{2\Delta^2} \tag{5.7}$$

for every ε-approximate solution x of (5.6). If, in addition, ε can be chosen so that

$$\log(1/\varepsilon) \text{ is polynomial in } n \text{ and } L, \tag{5.8}$$

then an ε-approximate solution x of (5.6) can be computed by the ellipsoid method in time polynomial in n and L. It follows from a derivation in [23] that one can specify ε as a function of n, L, and $||c||$ so that (5.7) and (5.8) are satisfied. With $\Delta = 2^L$ and $\gamma = 2^{Ln+L+1}$, one can set $1/\varepsilon = n^{3/2}\, 2^{(n^2+2n+2)L+2n+5} + ||c||\, n^{1/2}\, 2^{2nL+3L+5}$. Since $x_j < 2^L$ for each component x_j of x, the rounding of x by continued fractions requires at most $O[n(p+L)]$ arithmetic operations each involving numbers with at most $p+L$ binary digits, where p is the number of binary digits of precision maintained in the ellipsoid method.

6. Implementation

In our description of the ellipsoid algorithm, we have followed Gács and Lovász [14] in representing an ellipsoid E_k by its center x_k and a positive definite symmetric matrix B_k. This representation results in particularly simple updating formulae for determining x_{k+1} and B_{k+1}, and hence E_{k+1}. Unfortunately, however, if these formulae are used to implement the ellipsoid algorithm on any currently available finite precision computer, roundoff errors will almost invariably cause the computed matrices B_k to become indefinite. Consequently, the quantity $a^T B_k a$, whose square root is required, may turn out to be zero or negative.

To avoid such numerical difficulties one must implement the ellipsoid algorithm in a numerically stable way. One approach that can be formula (6.3) can result in J_k losing rank as a result of roundoff errors.

It is possible to keep J_k in (6.3) and L_k in (6.6) in product form. Indeed such implementations are analogous to the product form of the inverse and the Bartels-Golub LU factorization for the basis matrix in the simplex method [5].

7. Some Recent Work and Extensions

Although the ellipsoid algorithm was unknown to almost all of the mathematics-operations research-computer science community outside of the Soviet Union before the summer of 1979, the excitement generated by Khachiyan's note, due in part to its coverage in the popular press, has already resulted in a rather large number of technical papers on the subject. In this section we will try to summarize some of the more important results contained in this recent work, and point out promising avenues for further study.

In [14] Gács and Lovász described the ellipsoid algorithm as it is presented in Section 2 and supplied proofs for the claims in Khachiyan's note, ignoring some of his computational considerations for the purpose of exposition. With the dissemination of this report and its presentation at the X International Symposium on Mathematical Programming in Montreal, Canada, in August 1979, widespread investigation of the ellipsoid algorithm began in earnest.

Of the several papers that have appeared which treat the computational complexity of the ellipsoid algorithm in the spirit of Gács and Lovász [14], the papers by Padberg and Rao [47,48] are probably the most comprehensive. In [47] the validity of the ellipsoid algorithm with deep cuts, i.e. (2.5) and (2.6) with step, dilation, and expansion parameters defined by (4.1) and (4.2), is proved under the assumption that all computations are performed in exact arithmetic. These results are extended in [48] to the finite precision case. Padberg and Rao [47] also derive a polynomial bound for the computational effort required by the bisection method described in Section 5 for solving the linear programming problem (5.1). By far the most important theoretical advance that has come from the ellipsoid algorithm since Khachiyan's breakthrough is the paper by Grötschel, Lovász and Schrijver [23]. In [23] it is shown how the full generality of the ellipsoid algorithm can be used to devise polynomial-time algorithms for some combinatorial optimization problems and to prove that certain other problems are *NP*-hard. We illustrate this approach in some detail in the next section.

The Grötschel, Lovász, and Schrijver paper also deals with the accuracy required in performing calculations with the ellipsoid method, and it proves that the sliding objective function approach for solving optimization problems and the method for rounding approximate solutions to obtain exact solutions described in Section 5 can be done in polynomial time. They show that if δ in (2.6) is replaced by $\frac{2n^2+3}{2n^2}$ --i.e., if the axes of E_{k+1} are expanded slightly--then the ellipsoid method remains valid if all calculations are rounded to a prescribed finite precision. Khachiyan [35] replaces $\delta^{1/2}$ in (6.5) by $2^{1/8n^2}\delta^{1/2}$ to achieve a similar result.

In [37], Kozlov, Tarasov and Khachiyan describe a polynomial-time algorithm based upon the ellipsoid method and the bisection method for solving convex quadratic programming problems. After obtaining an approximate optimal given by Krol and Mirman [38], both the matrix J_k and the constraint data A^k and b^k in the affine-transformed space are updated on each iteration.

Halfin also remarks that there is a similarity between the update of A^k--i.e., $a_j^{(k+1)} = \delta^{1/2}(a_j^{(k)} - (\pi a_p^{(k)^T} a_j^{(k)} / a_p^{(k)^T} a_p^{(k)}) a_p^{(k)})$, $j = 1,\ldots,m$--and a simplex pivot. Whether this observation will lead to some understanding of the relationship of the ellipsoid algorithm to the simplex method remains unclear. In fact, the above iterative formula appears to bear a closer resemblance to Gram-Schmidt orthogonalization ($\delta = \pi = 1$) than to a simplex pivot. Although this resemblance is not mentioned in [24], it is shown that $a_p^{(k)}$ becomes "more orthogonal" to all of the other $a_j^{(k)}$'s after a space dilation based upon $a_p^{(k)}$. However, in [21] it is observed that if $a_p^{(k)}$, $a_i^{(k)}$ and $a_j^{(k)}$ are mutually obtuse, then as a consequence of such an ellipsoid step $a_i^{(k)}$ and $a_j^{(k)}$ will become more obtuse, even though they become more orthogonal to $a_p^{(k)}$. For the special case of a system of linear equations Goffin [19] proves that $AB_{kn}A^T = (\delta^n(1-\sigma))^k(D+\Omega)$, where D is a positive definite diagonal matrix, Ω is a matrix with elements whose order of magnitude is $(1-\sigma)^k$, and the cuts are chosen cyclically. Thus in the metric corresponding to B_k the constraint normals progressively become more and more orthogonal.

Nearly one-half of the papers that have appeared since the Gács and Lovász paper [14] have been concerned with modifying the basic algorithm (2.5)-(2.7) in order to reduce the volume of E_k as much as possible. The most obvious way to do this is to use deep or possibly "deepest" cuts.

As already mentioned in Section 3, Iudin and Nemirovskii's [27] description of the ellipsoid algorithm allows for cuts that do not pass through the center of the ellipsoid. Although they were interested in "shallow" cuts, their formulae apply to deep cuts as well. Because the English translation of [27] was unknown to most researchers, much of the recent work on improving the ellipsoid algorithm has involved the rediscovery of these formulae.

Krol and Mirman [38] and Goffin [19] give relaxations of the deep-cut version of the ellipsoid method by allowing different formulae for τ, σ and δ. Krol and Mirman hypothesize that such a choice may result in a faster algorithm, since a locally optimal strategy for volume reduction is not necessarily globally optimal.

The idea of using surrogate cuts was proposed independently by Goldfarb and Todd [21] and Krol and Mirman [38]. The sufficient conditions for $\bar{u}$ in (4.6) to give a deepest surrogate cut are from [21]. Krol and Mirman [38] give necessary and sufficient conditions for forming a surrogate cut using a deepest cut together with another less violated--possibly even satisfied--constraint. These conditions indicate whether or not the $\bar{u}$ in (4.6) for the 2-constraint case is nonnegative. Since the surrogate cut is deeper than either of the cuts from which it is generated, the process can be repeated iteratively using the newly formed surrogate cut and a regular cut. If a valid surrogate cut cannot be formed, then either the point on the current deepest (surrogate) cut closest to the center of E_k in the metric B_k is a solution to the system of linear inequalities (2.1), or that system is infeasible. The iterative procedure described in [38] which is based upon these observations

Just as in the simplex method, there are many ways to implement the ellipsoid method. These include using and updating: (i) the positive definite matrix B_k as described in Section 2 [14], [47], [23]; (ii) the matrix J_k which transforms a sphere into the ellipsoid E_k translated to the origin, [53], [35], [38]; (iii) the Cholesky or LDL^T factorization of B_k [10], [21]; and (iv) the problem data under the transformation induced by J_k [24], [38]. A product form version of (iii) is discussed in [21]. One of the principal computational and practical drawbacks of the ellipsoid method is that it does not appear to be possible to implement it so as to take advantage of any sparsity in the problem data, other than block diagonal structure. To save work, it also has been suggested that the ellipsoid and relaxation methods be combined into a hybrid algorithm [21], [60]. If α is large enough one can simply scale B_k; i.e., set $\tau = \alpha$, $\delta = 1-\alpha^2$ and $\sigma = 0$ in (2.5) and (2.6). If $\alpha \geq 1/n$ the volume ratio is $< (1 - \frac{1}{n^2})^{n/2} < e^{-1/2n}$; hence such an algorithm is polynomial. A hybrid algorithm which combines the ellipsoid method with the simplex method is proposed in [68].

It is also important to mention that the ellipsoid method can be applied to problems other than systems of linear inequalities and linear programs. As already stated in Section 3, the ellipsoid method was developed for solving convex (not necessarily differentiable) optimization problems [53], [27]. Clearly much of our discussion in the preceding sections is applicable to this more general setting. More is said in the next section about the full generality of the method. It seems more likely that the ellipsoid method will be found to be of some practical value for nonlinear and nondifferentiable problems than for linear programming. Kozlov, Tarasov and Khachiyan [37] have used the ellipsoid method in conjunction with the method of bisection to show that the convex quadratic programming problem is in P. Several other authors have used it to attack the linear complementarity problem [1], [6], [32].

Finally, we note that although relatively little computational experience with the ellipsoid method has been reported, the general concensus is that at present it is not a practical alternative to the simplex method for linear programming problems. A list of papers that report such computational results appears in [64]. In fact the only mildly encouraging results are those reported by Krol and Mirman [36].* Our own computational experience indicates that the slow convergence exhibited in the example analyzed in Appendix C is rather typical. We found that in spite of using deepest cuts, surrogate cuts of several types, and other refinements, on the average the parameter α (see 4.2) was approximately $1/n$. Thus, unless there are some further breakthroughs in implementation, it seems unlikely that the ellipsoid method will replace the simplex method as the computational workhorse of linear programming.

8. Combinatorial Implications

It is intriguing that the overall approach of the ellipsoid method does not depend directly on the availability of an explicit list of the defining inequalities, or even on linearity. In a very interesting paper, Grötschel, Lovász, and Schrijver [23] examine the ellipsoid method in a general framework, establish theoretical results based on the general

*More recently at the Spring 1980 ORSA meeting, Krol and Mirman expressed pessimism about the practicality of the method.

problem is to design at minimum cost an undirected communication network with n nodes and prescribed two-terminal flow values. We denote the set of nodes by $N = \{1,\ldots,n\}$. In the n'-vector $x = (x_{ij})$ of nonnegative decision variables each component x_{ij} represents the capacity of the link $\{i,j\}$ between nodes i and $j \neq i$.* The network must have sufficient capacity to accommodate a flow rate of r_{ij} units between nodes i and $j \neq i$ when they act as the unique source-sink pair. The cost of providing capacity x_{ij} on link $\{i,j\}$ is $d_{ij} \cdot x_{ij}$, and the objective is to provide sufficient capacity to meet all n' requirements r_{ij} at minimum total cost $d^T x = \sum_{1 \leq i < j \leq n} d_{ij} x_{ij}$. Note that the decision variables are permitted to assume non-integer values, for example when $n = 3$ and $d = r = (1,1,1)^T$ the unique optimal solution has $x = (1/2, 1/2, 1/2)^T$.

In the special case where all $d_{ij} = 1$ (or d is constant over all links) Gomory and Hu (see [13]) give a beautifully simple procedure for solving the synthesis problem. Gomory and Hu [22] also point out that the general problem, though not solvable by their simple procedure, is at least a linear programming problem, which unfortunately has an enormous number of defining inequalities. From the Max-Flow Min-Cut Theorem of Ford and Fulkerson (see [13]) we know that a given $x \in \mathbb{R}^{n'}$ satisfies the single requirement r_{ij} if and only if the capacity of every i-j cutset is at least r_{ij}, i.e., if and only if for every $Y \subseteq N$ having $i \in Y$, $j \in \overline{Y} \equiv N \backslash Y$

$$x(Y,\overline{Y}) \equiv \sum_{\substack{h \in Y \\ k \in \overline{Y}}} x_{hk} \geq r_{ij}. \tag{8.2}$$

*In the discussion of this example we will denote the center of the kth ellipsoid by x^k rather than x_k, so that x^k_{ij} is the (i,j)-component of x^k.

Thus the set of all feasible solutions x of our synthesis problem can be described as the polyhedron P consisting of all $x \geq 0$ satisfying (8.2) for all $1 \leq i < j \leq n$. A large number of the conditions (8.2) can obviously be discarded. If $n \geq 3$ then for a given $\emptyset \neq Y \subsetneq N$ there will be different pairs $\{i,j\}$ and $\{i',j'\}$ such that $i,i' \in Y$ and $j,j' \in \overline{Y}$, so one of the constraints $x(Y,\overline{Y}) \geq r_{ij}$ and $x(Y,\overline{Y}) \geq r_{i'j'}$ is implied by the other. Hence we can write the network synthesis problem as the linear programming problem

$$\text{minimize} \quad d^T x \tag{8.3a}$$

$$\text{subject to} \quad x(Y,\overline{Y}) \geq r_Y \quad \text{for all} \quad \emptyset \neq Y \subsetneq N, \tag{8.3b}$$

$$x \geq 0 \tag{8.3c}$$

where $r_Y = \max\{r_{ij}: i \in Y,\ j \in \overline{Y}\}$. This still leaves us with $2^{n-1}-1$ distinct inequalities of the form (8.3b), each involving only $n' = \frac{1}{2}n(n-1)$ variables. Moreover all of the conditions (8.3b) having $r_Y > 0$ define facets of P; none can be deleted without properly relaxing the feasible set. To apply the ellipsoid method in the standard way directly to (8.3) would result in two overwhelming problems caused by the large number of inequalities (8.3b) relative to the size

$$\ell = \lfloor \log n \rfloor + \sum_{\substack{1 \leq i < j \leq n \\ r_{ij} \neq 0}} \lfloor \log r_{ij} \rfloor + \sum_{\substack{1 \leq i < j \leq n \\ d_{ij} \neq 0}} \lfloor \log d_{ij} \rfloor + 2(n^2-n+1)$$

of the problem encoding.

There are several polynomial time implementations of the maximum flow algorithm; the flow algorithm of [44] will solve each of our flow problems in $O(n^3)$ computations, each involving numbers with at most $\phi \leq \log(2R+2) + p$ binary digits, where p is the number of digits of accuracy maintained in the updates.

Based on the comments above it should now be evident that we can compute an ϵ-approximate optimal solution of (8.3) in time polynomial in ℓ and $\log(\frac{1}{\epsilon})$. We can then round our ϵ-approximate solution to an exact solution as described in Section 5. For this to be done in time polynomial in ℓ, we need to choose Δ and ϵ so that $\log(\frac{1}{\epsilon})$ is bounded by a polynomial in ℓ, (not L, the size of the encoding of the linear programming formulation (8.3) of the problem). That our linear programming basic solutions can arise from nonsingular systems with $2^{n-1}-1$ rows and columns looks discouraging. Note however that all but $q \leq n'$ basic columns are slack (unit) vectors. So the values assumed by the basic x_{ij}-variables arise from a $q \times q$ subsystem $Ax = b$, where A is a submatrix of the (0,1)-constraint matrix of (8.3). It follows that $\det A \leq [\frac{1}{2} n(n-1)]! < (n^2)! < n^{2n^2}$, permitting us to select $\Delta = n^{2n^2}$, which allows ϵ to be chosen so that $\log(\frac{1}{\epsilon})$ is polynomial in ℓ.* Thus we achieve a polynomial-time algorithm for the network synthesis problem.

Most combinatorial optimization problems can, like the network synthesis problem, be recast as linear programming problems in which the number of defining inequalities is exponential in the size of the original problem encoding (although it is usually very difficult to find

---

*One suitable choice is $\frac{1}{\epsilon} = 2^{n+4} n^{2n^4+4n^3+4n^2+n} + 2^5 ||d|| n^{4n^3+6n^2+1/2}$.

an explicit description of the defining inequalities). In many of these problems, including many *NP*-complete problems, one can specify values of x^0, R, ρ, p, and Δ that guarantee a polynomial bound on the number of computations performed in the optimization routine of the ellipsoid method. In the terminology of complexity theory, the ellipsoid algorithm provides a polynomial Turing reduction from the optimization problem to the separation problem (see [15]). To solve such problems in polynomial time it suffices to give a polynomial-time separation routine. Grötschel, Lovász, and Schrijver [23] have indicated how to accomplish this for a variety of problems including: optimum branchings, the undirected Chinese Postman Problem, minimum weight perfect matchings and maximum weight matchings in graphs, minimization of submodular set functions, and the stability number in perfect graphs. The latter two problems were not previously known to be in *P*. Though the others were known to be in *P*, the approach of [23] is new, and the ease with which it embraces such a variety of problems is rather provocative.

The casual reader should resist any temptation to conclude that one should be able to immediately show that $P = NP$ by exhibiting a polynomial-time separation routine for, say, the traveling salesman problem. Certainly one can use the approach of [23] to establish that separation problems associated with some *NP*-hard problems are also *NP*-hard, as in the example that follows. But it might be naive to expect it to be any easier to find a polynomial-time algorithm for such a separation problem than for the problems previously known to be *NP*-complete.

Optimal Extreme Points of Polyhedra

First consider the problem

$$\text{maximize } \{c^T x : x \in \mathrm{Ext}(P)\}, \tag{8.4}$$

network decomposes into a sum of directed s-t path flows is *NP*-hard; there exists a polynomial-time algorithm for this problem only if $P = NP$.

Polynomial Equivalence of Optimization and Separation

Our comments thus far have not fully portrayed the strength of the results of [23]. Suppose that $\mathcal{K}^*$ is a class of polytopes K each with known x^0 and $0 < \rho < R$ such that

$$S(x^0,\rho) \subseteq K \subseteq S(x^0,R). \tag{8.7}$$

We have observed so far that the existence of a polynomial separation algorithm for $\mathcal{K}$ implies the existence of a weak (i.e., ε-approximate) optimization algorithm that is polynomial in $\log R$, $\log(\frac{1}{\rho})$, $\log(\frac{1}{\varepsilon})$ and L, the size of the encoding. Grötschel, Lovász, and Schrijver [23] actually show that it suffices to have a polynomial algorithm for all $K \in \mathcal{K}$ for the weak separation problem:

> given $z \in \mathbb{R}^n$ either determine that there exists $y \in K$ such that $||z-y|| \leq \varepsilon$, or give a vector $\pi \in R^n$, $||\pi|| > 1$, such that $\pi^T z \geq \pi^T y - \varepsilon$, $\forall y \in K$.

Furthermore they establish the converse of this result, namely that if the optimization problem is weakly solvable for $\mathcal{K}$, then so is the separation problem.* Indeed they demonstrate the polynomial equivalence of weak separation and weak optimization for any class $\mathcal{K}$ of convex (not necessarily polyhedral) bodies satisfying (8.7), and they use this additional generality in solving the stability number problem for perfect

*The very recent paper of Karp and Papadimitriou also deals with the relationship between optimization and separation in general combinatorial optimization [illegible] graphs. Since Khachiyan's work was motivated by the work of Shor [53] and Iudin and Nemirovskii [27] in convex optimization, it should not be surprising that these results go beyond the polyhedral domain. However, the rounding arguments that establish the equivalence of weak and strong (exact) optimization depend on polyhedral structure.

Grötschel, Lovász, and Schrijver [23] give an interesting algorithmic application of their result that polynomial-time optimization algorithms yield polynomial-time separation algorithms. The well known greedy algorithm is a polynomial-time algorithm for the maximum weight independent set problem in matroids. Thus by [23] one gets a polynomial-time algorithm for the related separation problem. Now given k matroids our ability to solve the separation problems in each, immediately yields a polynomial-time separation algorithm for k-matroid intersections, and hence one for (fractional) maximum weight independent vectors in the intersection of k matroids. When $k = 2$ the vertices of the intersection of the two matroid polyhedra will be integer, and thus Grötschel, Lovász, and Schrijver provide an alternative to Edmonds' algorithm [11] for the (2-) matroid intersection problem.

As with the application of the ellipsoid method to general linear programming, one must be careful here not to confuse the lovely results of [23] concerning theoretical efficiency with practical considerations. Grötschel, Lovász, and Schrijver do not suggest that their polynomial-time combinatorial algorithms should be used, as is, in the practical solution of such problems. Even in the network synthesis example, which has a reasonably modest bound on the number of iterations and a very easy separation routine, the required number of bits of accuracy to obtain the itself to sensitivity analysis or to the addition or deletion of constraints or variables. However, once a problem has been solved by the ellipsoid method, one can readily obtain this information by conventional techniques.

It is important to point out that because of the limitations of finite precision arithmetic it is unlikely that any reasonable implementation of the method would be polynomial. Indeed some researchers have been so distressed by the presence of L, the length of the problem encoding, in the bound on the number of iterations, that they are unwilling to consider the algorithm to be polynomial even with full precision. The presence of L is certainly unpleasant from a practical point of view, but is perfectly natural to the accepted Turing machine model of computation. Also, note that L is bounded by $(mn+m+n)(2+\log \sigma)$, where σ is the magnitude of the largest number in the data. Even in such elementary polynomial-time algorithms as Dijkstra's $O(n^2)$ shortest path algorithm (see [40]), the total computational effort depends on $\log \sigma$, in that each of the $O(n^2)$ steps involves operations on numbers with as many as $1 + \log \sigma$ bits. Because $\log \sigma$ appears in the bound on the number of iterations in the ellipsoid method, the bound on its total computational effort is a function of $(\log \sigma)^2$. If $\log \sigma$ is well accepted, $(\log \sigma)^2$ should cause no great distress.

It should be clear from our discussion of [23] in Section 8 that the ellipsoid method is a powerful theoretical tool and a unifying element in the analysis of the computational complexity of combinatorial optimization problems. This is especially striking given the non-combinatorial nature of the method. One of the most important and long-lasting effects that Khachiyan's result may have is to expand our perspective of linear programming and related combinatorial problems. Given the extensive use of the simplex method, it is ironic that many fundamental questions concerning its computational behavior remain unanswered. Perhaps the excitement caused by the ellipsoid will generate further research in this area.

with the property that g does not grow too rapidly as a function of s. We might be displeased if the best possible guarantee g had $g(s+1) \geq 2g(s)$ for all s (or even $g(s+k) \geq \lambda g(s)$ for all $s \geq t$, where $\lambda > 1$, and k and t are positive integers) indicating exponential growth in the number of computations performed by A as problem size increases. A is said to be a <u>polynomial-bounded</u> or <u>polynomial-time</u> algorithm if there exists a polynomial function $g(s)$ satisfying (A.3). Problem Q is called <u>polynomially solvable</u> if there exists a polynomial-time algorithm for Q. The class of all polynomially solvable (decision) problems is denoted by P. Note that in the determination of whether a problem is polynomially solvable it does not matter whether we encode integers in their binary expansions, or decimal expansions, or expansion in any base b larger than 1, since such a change increases the length of the encoding by at most a factor of $\log b$.

Polynomial boundedness was proposed by Edmonds [10] and independently by Cobham [7] as a <u>theoretical</u> criterion for algorithmic efficiency, and has been widely studied: among the problems for which there are known polynomial-time algorithms are the assignment, shortest path, maximum flow, and minimum cost flow problems. There are a large number of classical optimization problems in operations research for which there is no known polynomial-time algorithm. These include the traveling salesman problem, integer linear programming, and various production scheduling problems. No one has managed to show that there exists no polynomial-time algorithm for these problems, but the theory of NP-completeness offers substantial evidence of their difficulty. It implies, roughly, that there exists a polynomial-time algorithm for, say, the traveling salesman problem if and only if every problem solvable by a polynomial-depth branch-and-bound algorithm is solvable by a polynomial time algorithm.

The decision form of each of the problems noted above is a member of the class NP. Informally we can regard NP to be the class of all (decision) problems solvable by a backtrack search (or branch-and-bound) algorithm for which the depth of the search tree and the number of computations at each node (subproblem) can be bounded by fixed polynomials in the size of the problem encoding. (So that for every 'yes' (feasible) instance, there is some sequence of branches that leads to a 'yes' answer in polynomial time.) NP includes a large number of well-known problems; indeed it should be clear that $P \subseteq NP$, since a polynomial-time algorithm is trivially a polynomial-depth backtrack search algorithm (that never backtracks). Of course the breadth of a polynomial-depth tree may grow exponentially, as unfortunately occurs in the obvious backtrack search algorithms for the traveling salesman problem, so we might well imagine that there could be problems in NP not in P. The question of whether $P \subsetneq NP$ or $P = NP$ is unresolved; it is often called the "biggest" open problem in theoretical computer science. Most researchers consider it very unlikely that $P = NP$.

Although the $P \stackrel{?}{=} NP$ question is unresolved, problems that must be in $NP \setminus P$, if $P \neq NP$, have been identified. Among these are the NP-complete problems first examined by S. Cook [8]. Essentially, Cook showed how to devise from a polynomial-depth backtrack search algorithm for any problem Q, a polynomial-time algorithm A_Q that transforms instances of Q into equivalent instances of the satisfiability problem in the propositional calculus. Every problem Q in NP is in this way polynomial-reducible to satisfiability, which is itself in NP. Hence satisfiability can be regarded to be as hard as any problem in NP; it is said to be complete in NP, or NP-complete. In particular a polynomial-time

Appendix B: Minimum Volume Ellipsoids

Here we show that the formulae given in (2.5)-(2.7) and (4.1)-(4.2) yield an ellipsoid E_{k+1} of minimum volume containing the appropriate part of E_k. Since affine transformations multiply volumes by a constant factor, we may assume E_k is the unit ball and $a \in R^n$ is a multiple of the first unit vector. We denote the jth unit vector by e_j, $j = 1,2,\ldots,n$.

Hence suppose

$$E = \{x \in \mathbb{R}^n \mid ||x|| \le 1\} \quad \text{and} \quad H = \{x \in \mathbb{R}^n \mid e_1^T x \le -\alpha\}$$

and consider the general ellipsoid

$$\begin{aligned} E_+ &= \{x \in \mathbb{R}^n \mid ||J^{-1}(x-x_0)|| \le 1\} \\ &= \{x \in \mathbb{R}^n \mid (x-x_0)^T B^{-1}(x-x_0) \le 1\} \qquad \text{(B.1)} \end{aligned}$$

where $B = JJ^T$. We will prove

Theorem B.1. If $-1/n \le \alpha < 1$, the minimum volume ellipsoid containing $E \cap H$ is E_+, where

$$x_0 = -\tau e_1 \quad \text{and} \quad B = \delta(I - \sigma e_1 e_1^T) \qquad \text{(B.2)}$$

and

$$\tau = \frac{1+n\alpha}{n+1}, \quad \sigma = \frac{2(1+n\alpha)}{(n+1)(1+\alpha)} \quad \text{and} \quad \delta = \frac{n^2}{n^2-1}(1-\alpha^2). \qquad \text{(B.3)}$$

The theorem will follow from Lemmas B.3 and B.4 below. First we need to prove

Proposition B.2. (Hadamard's Inequality) Let Y be an $n \times n$ nonsingular matrix. Then

$$|\det Y| \leq \prod_{j=1}^{n} ||Ye_j||$$

with equality if and only if the columns of Y are orthogonal.

Proof: Since Y is nonsingular $A = Y^TY$ is positive definite and has a Cholesky factorization $A = LL^T$ where L is lower triangular. By definition

$$a_{jj} = \sum_{i=1}^{j-1} \ell_{ji}^2 + \ell_{jj}^2. \qquad (B.4)$$

Thus

$$(\det Y)^2 = \det A = (\det L)^2 = \prod_{j=1}^{n} \ell_{jj}^2$$
$$\leq \prod_{j=1}^{n} a_{jj} = \prod_{j=1}^{n} ||Ye_j||^2.$$

Equality holds if and only if $\ell_{ji} = 0$ for all $j \neq i$ by (B.4), i.e., if and only if $A = Y^TY$ is diagonal or, equivalently, the columns of Y are orthogonal.

Now we come to the first lemma. Note that $E \cap H$ contains the $2n-1$ points $-e_1$ and $-\alpha e_1 \pm (1-\alpha^2)^{1/2} e_i$, $i = 2,\ldots,n$. Let $\gamma = (1-\alpha^2)^{1/2}$.

Lemma B.3. If $-1 < \alpha < 1$, the ellipsoid of minimum volume containing the points $-e_1$ and $-\alpha e_1 \pm \gamma e_i$, $i = 2,\ldots,n$, is E_+ given by (3.1)-(3.3).

Proof: Let $Y = J^{-1}$ with columns $y_1,y_2,\ldots,y_n$ and let $y_0 = J^{-1}x_0$. Suppose E_+ in (B.1) contains the specified points. Then

Proof: Let $x^T = (\xi_1,\ldots,\xi_n)$ and note that

$$B^{-1} = \mathrm{diag}\left(\frac{(n+1)^2}{n^2(1-\alpha)^2}, \frac{n^2-1}{n^2(1-\alpha^2)}, \ldots, \frac{n^2-1}{n^2(1-\alpha)^2}\right).$$

Hence

$$(x-x_0)^T B^{-1}(x-x_0)$$

$$= \frac{n^2-1}{n^2(1-\alpha^2)} ||x||^2 + \left(\frac{(n+1)^2}{n^2(1-\alpha)^2} - \frac{n^2-1}{n^2(1-\alpha^2)}\right)\xi_1^2 + \frac{2(n+1)(1+n\alpha)}{n^2(1-\alpha)^2} \xi_1 + \frac{(1+n\alpha)^2}{n^2(1-\alpha)^2}$$

$$= \frac{n^2-1}{n^2(1-\alpha^2)} (||x||^2-1) + \frac{2(n+1)(1+n\alpha)}{n^2(1-\alpha^2)(1-\alpha)} \xi_1^2 + \frac{2(n+1)(1+n\alpha)}{n^2(1-\alpha)^2} \xi_1 + \frac{2(n+1)(1+n\alpha)\alpha}{n^2(1-\alpha^2)(1-\alpha)} + 1$$

$$= \frac{n^2-1}{n^2(1-\alpha^2)} (||x||^2-1) + \frac{2(n+1)(1+n\alpha)}{n^2(1-\alpha^2)(1-\alpha)} (\xi_1+\alpha)(\xi_1+1) + 1.$$

Thus if $-1/n \leq \alpha < 1$, $-1 \leq \xi_1 \leq -\alpha$ and $||x|| \leq 1$, the above expression is at most one, and $x \in E_+$.

A more general version of the theorem, involving cuts by two parallel hyperplanes, can be found in Todd [62].

Appendix C: An Example

We describe an example where the iterates $\{x_k\}$ and $\{B_k\}$ can be given explicitly and which demonstrates that convergence can be very slow even when the deep (or even deepest) cuts of Section 4 are used. This example also shows that the iterates $\{x_k\}$ need not converge toward the feasible set if that set has zero volume. If the feasible set is empty, then even an infinite sequence of iterations employing deepest cuts will not necessarily reveal infeasibility.

We again use e_j for the jth unit vector, and denote the components of x (x_k) by ξ_j $(\xi_{k,j})$. Suppose we are trying to find $x \in \mathbb{R}^n$ satisfying

$$\xi_j \leq 0, \quad -\xi_j \leq 0 \quad \text{for} \quad j = 1,2,\ldots,n; \tag{C.1}$$

even though the solution set has zero volume, we will ignore the perturbations of Section 2.

Let us start with

$$x_0 = (1,1,\ldots,1)^T \quad \text{and} \quad B_0 = n^2 I.$$

The outward normals to the constraints are $\pm e_j$, $j = 1,2,\ldots,n$, and $(\pm e_j)^T B_0 (\pm e_j) = n^2$ for all i. Thus the α corresponding to each constraint is $\pm 1/n$.

Our algorithm chooses one of the violated constraints, say $\xi_1 \leq 0$, as the cut; thus from (4.1) $\tau = 2/(n+1)$, $\sigma = 4n/(n+1)^2$ and $\delta = 1$. It follows that

References

[1] I. Adler, R.P. McLean, and J.S. Provan, "An Application of the Khachian-Shor Algorithm to a Class of Linear Complementarity Problems," undated (received April 1980), Cowles Foundation Discussion Paper No. 549, Cowles Foundation for Research in Economics, Box 2125, Yale Station, New Haven, CT 06520, USA.

[2] S. Agmon, "The Relaxation Method for Linear Inequalities," Canadian Jounral of Mathematics 6 (1954), 382-392.

[3] A.V. Aho, J.E. Hopcroft, and J.D. Ullman, The Design and Analysis of Computer Algorithms, Addison-Wesley, Reading, Mass. (1976).

[4] D. Avis and V. Chvátal, "Notes on Bland's Pivoting Rule," Math. Programming Studies 8 (1976), 24-34.

[5] R.H. Bartels, "A Stabilization of the Simplex Method," Numer. Math. 16 (1971), 414-434.

[6] S.J. Chung and K.G. Murty, "A Polynomially Bounded Algorithm for Positive Definite Symmetric LCPs," December 1979, Technical Report No. 79-10, Department of Industrial and Operations Engineering, The University of Michigan, Ann Arbor, MI 48109, USA.

[7] A. Cobham, "The Intrinsic Computational Difficulty of Functions," in Y. Bar-Hillel (ed.) Proc. [illegible] International Congress for Logic, Methodology, and Philosophy of Science, North-Holland, Amsterdam (1965), 24-30.

[8] S.A. Cook, "The Complexity of Theorem-Proving Procedures," Proc. ACM Symp. Theory of Computing 3 (1971), 151-158.

[9] G.B. Dantzig, Linear Programming and Extensions, Princeton University Press, Princeton, NJ (1963).

[10] J. Edmonds, "Paths, Trees and Flowers," Canad. J. Math. 17 (1965), 449-467.

[11] J. Edmonds, "Matroid Partition," Lectures in Applied Math 77, American Math. Soc. Providence, R.I. (1967), 335-346.

[12] Iu. M. Ermolev, "Methods of Solution of Nonlinear Extremal Problems," Kibernetika 2(4) (1966), 1-17, translated in Cybernetics 2(4) (1966), 1-14.

[13] L.R. Ford, Jr. and D.R. Fulkerson, Flows in Networks, Princeton University Press, Princeton, NJ ([illegible]).

[14] P. Gács and L. Lovász, "Khachiyan's Algorithm for Linear Programming," Technical Report STAN-CS-79-75 , Computer Science Department, Stanford University, California (1979).

[15] M.R. Garey and D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman, San Francisco (1979).

[16] P.E. Gill, W. Murray and M.A. Saunders, "Methods for Computing and Modifying the LDV Factors of a Matrix," Mathematics of Computation 29 (1975), 1051-1077.

[17] J.-L. Goffin, "Acceleration in the Relaxation Method for Linear Inequalities and Subgradient Optimization," Working Paper 7[illegible]-1[illegible], Faculty of Management, McGill University, Montreal, Canada, to appear in the proceedings of a task force on nondifferentiable optimization held at IIASA, Laxenburg, Austria, December 1978.

[18] J.-L. Goffin, "On the non-polynomiality of the relaxation method for system of inequalities," November 1979, Faculty of Management McGill University, 1001 Sherbrooke St. W., Montreal, Quebec, Canada H3A 1G5.

[19] J.-L. Goffin, "Convergence of a Cyclic Shor-Khachian Method for Systems of Linear Equalities," December 1979, Working paper No. 79-[illegible], Faculty of Management, McGill University, 1001 Sherbrooke St., W., Montreal, Quebec, Canada H3A 1G5.

[20] D. Goldfarb and W.Y. Sit, "Worst Case Behavior of the Steepest Edge Simplex Method," Discrete Applied Math 1 (1979), 277-285.

[21] D. Goldfarb and M.J. Todd, "Modifications and Implementation of the Shor-Khachian Algorithm for Linear Programming," January 1980, Technical Report 406, Department of Computer Science, Cornell University, Ithaca, N.Y. 14853, U.S.A.

[22] R.E. Gomory and T.C. Hu, "An Application of Generalized Linear Programming to Network Flows," SIAM J. Appl. Math. 10 (1962), 260-283.

[23] M. Grötschel, L. Lovász, and A. Schrijver, "The Ellipsoid Method and Its Consequences in Combinatorial Optimization," Combinatorica, to appear.

[24] S. Halfin, "The Sphere Method for Khachiyan's Algorithm," [illegible] (received March 1980), Bell Telephone Laboratories, Holmdel, N.J. 07733, U.S.A.

[25] A.J. Hoffman, "On Approximate Solutions of Systems of Linear Inequalities," J. Res. Natl. Bur. Standards 49 (1952), 263-265.

[54] N.Z. Shor, "Convergence Rate of the Gradient Descent Method with Dilatation of the Space," Kibernetika 6(2) (1970), 80-85, translated in Cybernetics 6(2) (1970), 102-108.

[55] N.Z. Shor, "Generalized Gradient Methods of Nondifferentiable Function Minimization and Their Application to Problems of Mathematical Programming," Ekonomika i Matematicheskie Metody 12 (1976), 337-356.

[56] N.Z. Shor, "Cut-off Method with Space Extension in Convex Programming Problems," Kibernetika 13(1) (1977), 94-95, translated in Cybernetics 13(1) (1977), 94-96.

[57] N.Z. Shor, "New Development Trends in Nondifferentiable Optimization," Kibernetika 13(6) (1977), 87-91 translated in Cybernetics 13(6) (1977), 881-886.

[58] N.Z. Shor and N.G. Zhurbenko, "A Minimization Method Utilizing the Operation of Space Expansion in the Direction of the Difference of Two Successive Gradients, Kibernetika 7(3) (1971).

[59] V.A. Skokov, "Note on Minimization Methods Employing Space Stretching," Kibernetika 4(4) (1974), 115-117 translated as Cybernetics 4(4) (1974), 689-692.

[60] J. Telgen, "On Relaxation Methods for Systems of Linear Inequalities," Technical Report, University of Tenn., Management Science Program, January 1980.

[61] M.J. Todd, "Some Remarks on the Relaxation Method for Linear Inequalities," Technical Report 419, School of Operations Research and Industrial Engineering, Cornell University, Ithaca, New York (1979).

[62] M.J. Todd, "Minimum Volume Ellipsoid Containing Part of a Given Ellipsoid," Technical Report No. 468, School of Operations Research and Industrial Engineering, Cornell University, Ithaca, New York (1980).

[63] P. Wolfe, "A Method of Conjugate Subgradients for Minimizing Nondifferentiable Functions," Math. Programming Study 3 (1975), 145-173.

[64] P. Wolfe, "A Bibliography for the Ellipsoid Algorithm," IBM Research Center Report, July 7, 1980.

[65] N. Zadeh, "A Bad Network Problem for the Simplex Method and Other Minimum Cost Flow Algorithms," Math. Programming 5 (1973), 255-266.

[66] R.M. Karp and C. H. Papadimitriou, "On Linear Characterizations of Combinatorial Optimization Problems."

[67] M.W. Padberg and M.R. Rao, "The Russian Method and Integer Programming," GBA Working Paper, NYU, Jan. 19[illegible].

[68] P.F. Pickel, "Some Improvements to Khachiyan's Algorithm in Linear [illegible]