Feature Article

The Ellipsoid Method: A Survey

ROBERT G. BLAND, DONALD GOLDFARB and MICHAEL J. TODD

Cornell University, Ithaca, New York

(Received August 1980; accepted July 1981)

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, but 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 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 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 in informal discussions. By the end of 1979 Khachiyan's note had become front-page news, not only for researchers, but for readers of major daily newspapers in the United States, Europe, and Japan (see Wolfe [1980]).

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

Subject classification: 660 ellipsoid method.

Operations Research
Vol. 29, No. 6, November–December 1981

0030-364X/81/1039-2906 $01.25

with nonnegative arc lengths), or to be $\mathcal{NP}$-complete (e.g., the traveling salesman problem, and the shortest path problem with arbitrary arc lengths). (Appendix A provides an informal discussion of the notions of polynomial boundedness and $\mathcal{NP}$-completeness for the unacquainted reader. For a rigorous treatment see Aho et al. [1976], Garey and Johnson [1979], Karp [1972, 1975].) 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 $\mathcal{NP}$-complete problems, but no one had managed to prove its membership in $\mathcal{P}$, the class of problems solvable by polynomial-time algorithms. Finally, Khachiyan indicated how one could adapt the ellipsoid method for convex optimization developed by the Soviet mathematicians N. Z. Shor, D. B. Iudin, and A. S. Nemirovskii to give a polynomial-time algorithm for linear programming. This algorithm differs dramatically from the simplex method: it is not a pivoting method; it uses metrical properties of $\mathbb{R}^n$; and it does not depend directly upon linearity of the objective function or the constraints.

Proofs of the claims in Khachiyan's note were provided by Gács and Lovász [1981] under the assumption of exact arithmetic, for ease of exposition. (In a more recent paper Khachiyan [1980] gives proofs of his earlier claims.) The presentation by Gacs and Lovasz at the International Symposium on Mathematical Programming in Montreal in August of 1979 began a widespread investigation of the ellipsoid method.

The Ensuing Commotion

The resolution of this major theoretical question concerning linear programming resulted in great (and deserved) excitement among researchers. The ensuing commotion in the popular press resulted from an unusual combination of circumstances. The great importance of linear programming and the simplex method (see Dantzig [1963]) as decision-making tools in government and industry led people to conclude correctly that a major practical improvement in our ability to solve linear programming problems could have substantial impact. In spite of disclaimers from theoretical researchers, journalists inferred that the theoretical efficiency (polynomial-boundedness) of the ellipsoid method must immediately translate into a major practical advance: "Shazam!" exclaimed the *New York Times* (November 11, 1979). The initial articles on Khachiyan's paper in some of the popular science magazines were mostly accurate, though some were potentially misleading. As those stories were digested (indigested?) by newspaper writers, the tale became so distorted that one familiar with Khachiyan's note could have legitimately wondered whether some of the newspaper articles were discussing a different paper by him. These articles characterized the new result as a profound break-

through in the solution of real-world problems, went on to suggest that the work was likely to result in efficient new algorithms for the traveling salesman problem, and ultimately some even declared the traveling salesman problem well-solved. For details concerning these accounts see the note by Lawler [1980], who compares the treatment of Khachiyan's paper in the popular press to the children's whispering game, "telephone."

Theoretical vs. Practical Considerations

In order to develop means for formal comparisons of algorithms and of problems, the theory of computational complexity builds upon certain notions such as polynomial boundedness. These ideas have produced a rich theory that has also yielded some important practical advances in algorithms. However an algorithm that is superior according to theoretical criteria is not necessarily superior in practice.

Suppose algorithms $\mathscr{A}$ and $\mathscr{A}_p$ solve problem $\mathscr{Q}$, $\mathscr{A}_p$ is polynomial-time, and $\mathscr{A}$ is not. Then there is some family $\{Q_n\}$ of instances of $\mathscr{Q}$ such that the running time of $\mathscr{A}$ on $\{Q_n\}$ increases faster than any polynomial function of n, while the running time of $\mathscr{A}_p$ on $\{Q_n\}$ is bounded by some polynomial function $f(n)$. For "large enough" values of n, $\mathscr{A}_p$ is guaranteed to run faster on Q_n than $\mathscr{A}$, and as n grows the discrepancy increases rapidly. But this is an asymptotic result; how large is "large enough"? For any positive integer n', one can easily construct a function g that is, say exponential in n, and such that $f(n) > g(n)$ for all $n \leq n'$. Thus it is possible that our nonpolynomial algorithm $\mathscr{A}$ might be preferable to the polynomial algorithm $\mathscr{A}_p$ for all instances of $\mathscr{Q}$ of the size that we expect to encounter in practice, although we must take care that our expectations are not too modest. Certainly before accepting a polynomial-time algorithm $\mathscr{A}_p$ as a useful practical tool, we would at least want to examine the particular polynomial function that bounds its computational performance. For example the known polynomial bound on the Dijkstra algorithm for computing shortest paths in directed graphs with nonnegative arc lengths and n nodes (see Lawler [1976]), is a small multiple 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 Khachiyan's note. 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 central sections of Levin and Newman 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 to increase its rate of convergence. In Section 4 three modifications to the basic algorithm are described. These are the use of deep cuts (i.e., violated inequalities), surrogate cuts (i.e., valid cuts formed by combining several inequalities), and parallel cuts (i.e., the use of two parallel inequalities,

one of which is violated). The numerical implementation of the ellipsoid method is considered in Section 6. Although several possibilities are discussed, we recommend the use of a Cholesky or LDL^T factorization of the positive definite matrix which determines the metric corresponding to each ellipsoid.

Section 5 describes three methods to adapt the ellipsoid algorithm of Section 2 to solve linear programming problems. The first method combines the primal and dual constraints and the weak duality inequality with its sense reversed. The second uses bisection to find the optimal value, while the third (closely related to the algorithms of Iudin and Nemirovskii, and Shor) is a sliding objective function method. A final subsection outlines how (in polynomial time) an exact solution to the linear programming problem can be obtained from the approximate one produced by the ellipsoid algorithm.

In Section 7 we discuss the relationship of the ellipsoid algorithm to other methods including the simplex method. Extensions to convex optimization and linear complementarity problems are also cited.

Section 8 concerns combinatorial applications of the ellipsoid method from the point of view of Grötschel et al. [1981]. This exceptionally interesting paper examines the ellipsoid method in greater generality, establishes theoretical results based on the general form of the algorithm, and uses those results to develop polynomial-time algorithms for a number of combinatorial optimization problems. We illustrate the approach of Grötschel, Lovász, and Schrijver in the context of examples familiar to many operations researchers, and relate it to the technique of column generation.

Section 9 consists of a few brief concluding remarks.

Three appendices are included. In the first we present, rather informally, some of the main ideas of computational complexity, including discussion of polynomial solvability and the classes of problems $\mathcal{P}$ and $\mathcal{NP}$. In the second appendix we give a proof that the ellipsoid constructed by the deep cut version of the ellipsoid method has the smallest volume among all ellipsoids containing the portion of the current ellipsoid on the appropriate side of that cut. In the third appendix we present an example which shows that convergence of the ellipsoid algorithm can be extremely slow, even if deep cuts are used. This example also demonstrates that the ellipsoid method, without the modifications introduced by Khachiyan, may converge to an infeasible point when applied to a system of inequalities whose solution set is nonempty, but not full-dimensional.

The paper is designed so that fairly complete coverage of essential aspects of the algorithm can be obtained without reading every section. The reader interested merely in the basic form of the algorithm and its application to linear programming should read Sections 2 and 5 (possibly omitting the last subsection) and the concluding remarks.

2. THE ELLIPSOID ALGORITHM

In this section we describe the ellipsoid method for determining the feasibility of a system of linear inequalities, and outline arguments that establish that the method can be made polynomial. We follow the interpretation by Gács and Lovász of Khachiyan's arguments.

Suppose we wish to find an n-vector x satisfying

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

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 $a_1, a_2, \cdots, a_m$, and the components of b are denoted $\beta_1, \beta_2, \cdots, \beta_m$. Thus (2.1) can be restated as

$$a_i^T x \leq \beta_i, \qquad i = 1, 2, \cdots, m.$$

We assume throughout that n is greater than one.

The Basic Iteration

The ellipsoid method constructs a sequence of ellipsoids $E_0, E_1, \cdots, E_k, \cdots$, 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

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

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). In essence Khachiyan showed that one can determine whether (2.1) is feasible or not within a prespecified (polynomial) 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

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

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 formulas

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

and

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

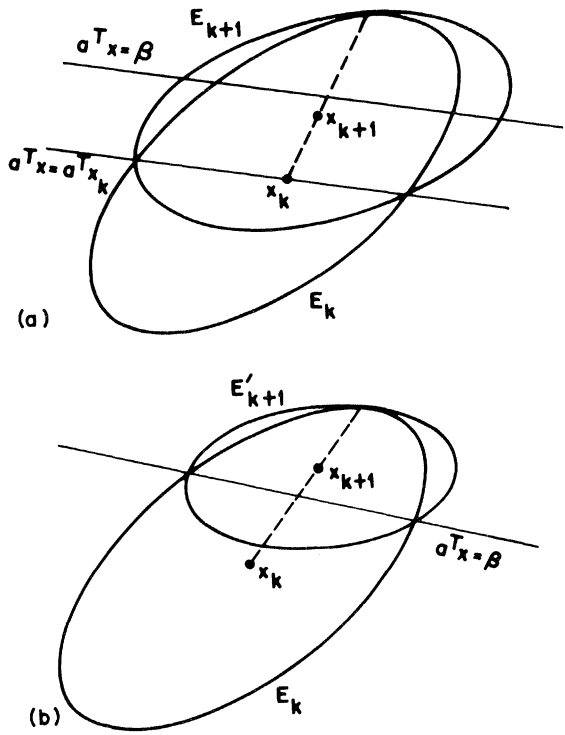

Figure 1. The ellipsoid method: (a) without deep cuts, (b) with deep cuts.

where

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

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 the identity so that E_k is a ball, then E_{k+1} is shrunk in the direction a by the factor

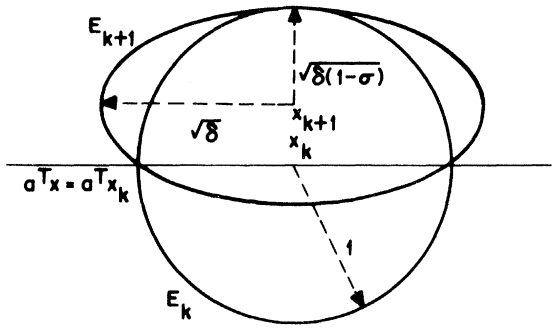

Figure 2. Geometric interpretation of the parameters.

$\sqrt{\delta(1-\sigma)} = n/(n+1)$ and expanded in all orthogonal directions by the factor $\sqrt{\delta} = n/\sqrt{n^2-1}$. (See Figure 2.)

It is intuitively clear that a smaller ellipsoid E'_{k+1} can be employed since it is only necessary that the new ellipsoid contain the section of E_k,

$$\{x \in E_k | a^T x \leq \beta\}, \tag{2.8}$$

rather the entire half-ellipsoid (2.3). (See Figure 1 (b).) This is indeed true, but we defer discussion of these "deep" cuts until Section 4.

Note that if E_k is a ball, then it is also possible to construct a ball S that contains the set (2.8) and is smaller in volume than E_k. Such a ball can have its center x_{k+1} on the open line segment $(x_k, x_k + 2(\hat{x}_k - x_k))$, where $\hat{x}_k$ is the projection of x_k onto the hyperplane $\{x \in \mathbb{R}^n | a^T x = \beta\}$. (See Figure 3.) The ball S will be smallest if x_{k+1} is $\hat{x}_k$. As we shall see in the next section such "ball" methods are well-known and predate the ellipsoid method.

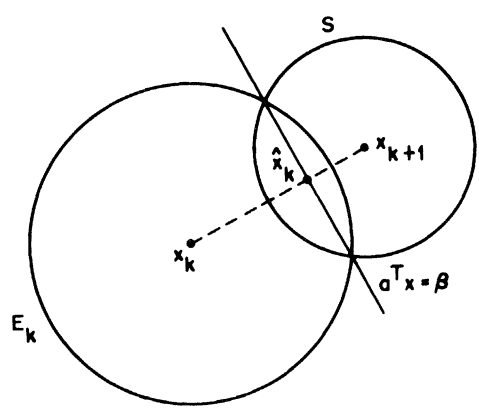

Figure 3. The ball method.

Polynomial Solvability

We now outline arguments that can be used to prove that the feasibility or infeasibility of (2.1) with integer data can be determined in polynomial time. For proofs see Gács and Lovász, or, for greater detail, see Padberg and Rao [1980a]. These analyses are based on the simplifying assumption that all computations are performed in exact arithmetic. In Padberg and Rao [1980b], Grötschel et al., and Khachiyan's recent paper [1980], proofs are supplied for the case of finite precision arithmetic.

We assume that the entries of A and b are integers. Then the length L of the input can be described in terms of the number of symbols required to encode it, i.e.,

$$L = \sum_{1\le i\le m,\,1\le j\le n,\,a_{ij}\neq 0} \lfloor \log|a_{ij}| \rfloor + \sum_{1\le i\le m,\,\beta_i\neq 0} \lfloor \log|\beta_i| \rfloor + \lfloor \log n \rfloor + \lfloor \log m \rfloor + 2mn + 2m + 4. \tag{2.9}$$

(See the explanation of (A.2) in Appendix A. The encoding involves four distinct symbols: +, −, 0, and 1; so the actual number of bits required is $2L$. Henceforth we take the liberty of using the term "bit" interchangeably with "symbol.")

We need to show that the number of steps is polynomial in L. There are two main issues in the proof. First, the formulas (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. Khachiyan uses the factor $2^{1/4n^2}$; Grötschel et al. replace δ in (2.6) by $(2n^2 + 3)/2n^2$ to achieve the same effect. 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 $S(y, \rho)$ denotes the ball of radius ρ centered at y. In this case we initialize with $E_0 = S(a_0, R)$. We now use the fact that when the formulas (2.4)–(2.7) are employed,

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

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

$$P \cap S(0, 2^L) \neq \emptyset; \tag{2.12}$$

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. Therefore let us perturb (2.1) to obtain the system of inequalities

$$2^L a_i{}^T x \leq 2^L \beta_i + 1, \qquad i = 1, 2, \cdots, m \tag{2.13}$$

with solution set P'. Khachiyan 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 of (2.13) (one method for obtaining exact solutions from approximate solutions is described at the end of Section 5), and the number of bits needed to represent (2.13) is at most $(m(n+1)+1)L$, hence polynomial in L. The reason for considering (2.13) is that if (2.1) has a feasible solution, say $\hat{x}$, then $S(\hat{x}, 1/\max_i \|2^L a_i\|)$ is contained in P'. Since $\|a_i\| \leq 2^L$, we obtain

$$S(\hat{x}, 2^{-2L}) \subseteq P' \tag{2.14}$$

if $\hat{x} \in P$. Note that the bound in (2.12) can be improved—certainly P contains a point within $2^L - 2^{-2L}$ of the origin if it is nonempty, and we can choose such a point for $\hat{x}$. Thus, if (2.1) is feasible,

$$S(\hat{x}, 2^{-2L}) \subseteq P' \cap S(0, 2^L) \subseteq S(0, 2^L). \tag{2.15}$$

Now by applying the arguments given above, if (2.1) is feasible and if the ellipsoid algorithm is applied to (2.13), then it must terminate with a feasible solution within $2n(n+1)\log(2^L/2^{-2L}) = 6n(n+1)L$ iterations. Hence if it fails to terminate in this many iterations, we can conclude that (2.1) is infeasible.

It has been convenient for exposition of the arguments above to apply the ellipsoid method to the perturbed system (2.13), as in Gács and Lovász. However, it is not necessary to implement the ellipsoid method

in this way; indeed, Khachiyan's approach, although essentially the same, does not explicitly perturb the original system (2.1).

A constraint $2^L a_i^T x \le 2^L \beta_i + 1$ of (2.13) is satisfied at the current iterate x_k if and only if the corresponding constraint $a_i^T x \le \beta_i$ of (2.1) has residual $a_i^T x - \beta_i \le 2^{-L}$. The procedure outlined above in terms of (2.13) can be described in the context of the original system (2.1) by merely requiring that the next cut be chosen from the constraints having residual larger than 2^{-L} at x_k. The half-ellipsoid (2.3) is the same whether the cut $a^T x \le a^T x_k$ is viewed as $a_i^T x \le a_i^T x_k$ from (2.1) or $2^L a_i^T x \le 2^L a_i^T x_k$ from (2.13); hence E_{k+1} and x_{k+1} are also the same. If the iterate x_k has maximum residual $\theta(x_k) \equiv \max_i a_i^T x_k - \beta_i \le 2^{-L}$, then x_k satisfies (2.13), and, as noted above, (2.1) must also have a solution.

Khachiyan works directly in terms of (2.1), using the condition $\theta(x_k) \le 2^{-L}$ to test for feasibility of (2.1). He generates the next cut from the constraint having maximum residual $\theta(x_k)$, which must be larger than 2^{-L} if feasibility has not been detected. So Khachiyan's condition for recognizing feasibility of (2.1) and his choice of cuts implicitly test whether x_k satisfies (2.13), and, if not, then selects the next cut to correspond to a constaint $2^L a_i^T x \le 2^L \beta + 1$ of (2.13) violated at x_k. Therefore the proof above that at most $6n(n+1)L$ iterations are required when cuts are generated from (arbitrary) violated constraints of (2.13) implies the same polynomial bound for Khachiyan's choice of a cut from an inequality with maximum residual (again assuming exact arithmetic).

3. HISTORICAL DEVELOPMENT

The publicity that arose from Khachiyan's announcement that a polynomial algorithm exists for linear programming has tended to blur the facts that the basic ellipsoid algorithm is not due to Khachiyan and arose in connection with convex, rather than linear, programming. In this section we describe the development of the method and give some perspective on its place in mathematical programming. We shall see that it is closely related to three earlier methods, two of which are quite well known. These antecedents are the relaxation method for linear inequalities, the subgradient method and the method of central sections for convex minimization.

The Relaxation Method

Relaxation algorithms for inequalities were introduced simultaneously by Agmon [1954] and Motzkin and Schoenberg [1954]. For the problem (2.1) they produce a sequence $\{x_k\}$ of iterates. At iteration $k+1$, if x_k is feasible the algorithm terminates; otherwise a violated constraint, say (2.2), is chosen and we set

$$x_{k+1} = x_k - \lambda_k a(a^T x_k - \beta)/a^T a, \tag{3.1}$$

where $\lambda_k = 2$ in Motzkin and Schoenberg's method and $0 < \lambda_k < 2$ in Agmon's method. The choice $\lambda_k = 1$ corresponds to projection of x_k onto the hyperplane $\{x \in \mathbb{R}^n | a^T x = \beta\}$. It can be seen that this method, with $0 < \lambda_k < 2$, corresponds to the "ball" method of Section 2—see Figure 3. Of course, it is unnecessary to have an a priori bound to define E_0 in order to implement the algorithm; however, if such a bound were available it would be straightforward to define a corresponding sequence $\{E_k\}$ of balls. Agmon showed that if (2.1) is feasible, if at each iteration the chosen constraint (2.2) is most violated in the sense of the Euclidean norm, and if λ_k is bounded away from 0 and 2, then his method converges linearly (i.e., at the rate of a geometrical progression). Indeed, he showed that each iterate comes closer by some fixed ratio to the set of feasible solutions than its predecessor. This ratio translates into a bound on the ratio of the volumes of the balls E_{k+1} and E_k. The main difference from the ellipsoid method is that this ratio depends on the data of the problem rather than just the dimension. Bounds on the ratio have been provided by Agmon, Hoffman [1952], Goffin [1978] and Todd [1979]. Todd [1979] and Goffin [1979a] demonstrate that an exponential (in the data) number of steps may be required.

The Subgradient and Space Dilation Methods

The subgradient method for minimizing a convex, not necessarily differentiable, function $f: \mathbb{R}^n \to \mathbb{R}$ was, apparently, first introduced by Shor [1964]. 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

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

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 as a special case a version of (3.1) in which a constraint with largest residual is chosen. Ermolev [1966] and Polyak [1967] give choices for μ_k that ensure global convergence; for example, $\mu_k \to 0$ and $\sum \mu_k = \infty$ suffice. However very slow convergence results. Polyak [1969] and Shor [1968] 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 [1970a, b] was 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

$$\begin{aligned} \tilde{g}_k &= J_k{}^T g_k / \|J_k{}^T g_k\|, \\ x_{k+1} &= x_k - \alpha_k J_k \tilde{g}_k, \\ J_{k+1} &= J_k(I - \beta_k \tilde{g}_k \tilde{g}_k{}^T), \end{aligned} \tag{3.4}$$

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 [1970b] describes precisely the difficulties with the linear convergence rate of his earlier subgradient method [1968]. 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 (see Shor [1970a]). For this algorithm, the minimum value of f must be known; Shor's later method [1970b] relaxes this requirement. Shor and Zhurbenko [1971] 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 Method of Central Sections

A third method on which the ellipsoid algorithm is based is that developed independently by Levin [1965] and Newman [1965], who addressed the problem of minimizing a convex function f over a bounded polyhedron $P_0 \subseteq \mathbb{R}^n$. 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

$$P_{k+1} = \{x \in P_k \mid g_k{}^T x \leq g_k{}^T x_k\},$$

where again g_k is a subgradient of f at x_k. Since f is convex, P_{k+1} contains all points of P_k whose objective function value is no greater than that of x_k. In this case, the volume of P_{k+1} is at most $(1 - e^{-1})$ times that of P_k. However, calculating the centers of gravity of polytopes with many facets in high dimensional spaces is an almost insuperable task. Levin proposed some simplifications for $n = 2$.

The Ellipsoid Method

The ellipsoid method was first described, rather cryptically, in a paper of Iudin and Nemirovskii [1976b]. In two papers [1976a, b] they discuss

the question of the "computational complexity" of convex programming problems: given a limited number of function and/or subgradient calls, with unlimited side calculations, how close can one get to the optimal value? To obtain upper bounds on such a deviation from optimality, specific methods must be proposed. Iudin and Nemirovskii [1976a] use a variant (the method of centered cross-sections) of the method of Levin and Newman; the burdensome "side calculations" of the centers of gravity are not counted in their analysis. For problems with parallelepipeds as their feasible regions, this method uses only a constant factor more iterations than an optimal method to obtain a given quality of solution. In their second paper, Iudin and Nemirovskii [1976b] discuss the computational difficulties of Levin's method and describe the modified method of centered cross-sections, using ellipsoids instead of polyhedra. The modified method is described for minimizing a convex function f with convex constraints; however, for the unconstrained problem of minimizing f in (3.3) it becomes the ellipsoid method of Section 2. This modified method may take n times as many iterations as the unmodified method to obtain a given quality of solution, but it is computationally implementable. Iudin and Nemirovskii describe the ellipsoid method implicitly in terms of a sequence $\{0_k\}$ of systems of coordinates. They also point out the rather surprising fact that the ellipsoid method is a special case of Shor's algorithm (3.4) with space dilation in the direction of the subgradient, when the parameters α_k and β_k are suitably chosen (in fact, $\alpha_k = \delta^{k/2}\tau$, $\beta_k = 1 - (1 - \sigma)^{1/2}$). Shor [1977a] gives the first completely explicit statement of the ellipsoid method as we know it.

As we have seen, one of the keys to the ellipsoid method is the ratio of volumes of successive ellipsoids. Iudin and Nemirovskii [1976b] state a geometrical lemma concerning an ellipsoid containing that portion of the unit ball in $\mathbb{R}^n$ with $a^T x \leq \cos \phi \|x\|$, where $\|a\| = 1$ and $0 \leq \phi \leq \pi/2$. If $\phi = \pi/2$, this part is precisely half the ball and the new ellipsoid is that given in Section 2. For $\phi < \pi/2$, it is the unit ball with a pointed circular cone cut out, and the new ellipsoid contains more than half the unit ball. Thus Iudin and Nemirovskii consider cuts that are "shallower" than cuts through the center but not cuts that are "deeper." (See Figure 1 (b) for a deep cut; these will be discussed in Section 4.) However, their formulas are valid for $\phi < 0$ (i.e. deep cuts) also. (They used the shallower cuts in algorithms where subgradients are not available and must be approximated by using function values. By cutting off less than half the current ellipsoid they could still guarantee that the desired solution was contained in the new ellipsoid in spite of the use of approximate subgradients.)

Khachiyan's [1979] contributions 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. For surveys on nondifferentiable optimization see Shor [1976, 1977b] and Polyak [1978]. Shor [1977b] states that computational comparisons of subgradient algorithms, subgradient algorithms with space dilation in the direction of the subgradient (Shor [1970a, b]) or in the direction of the difference of successive subgradients (Shor and Zhurbenko) and the "conjugate subgradient" methods of Lemarechal [1975] and Wolfe [1975] tend to favor the method of Shor and Zhurbenko, 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 by Nemirovskii and Iudin [1977] 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 [1970a, b], Skokov [1974] suggested updating the symmetric matrix $B_k = J_k J_k^T$ to save storage and reduce computation. His formulas are analogous to those of Gács and Lovasz 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. The most obvious way to do this is to use deep or, possibly, "deepest" cuts at each iteration to generate smaller ellipsoids. As already mentioned in Section 3, Iudin and Nemirovskii's [1976b] 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 formulas apply to deep cuts as well. Shor and Gershovich [1979] were the first to propose the use of deep cuts to speed up the ellipsoid method. Because both of these papers were unknown to most researchers, much of the recent work on improving the ellipsoid algorithm has involved the rediscovery of their formulas.

Deep Cuts

As before, suppose that x_k violates constraint (2.2). The ellipsoid E_{k+1} determined by formulas (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

Figure 1 (a) and (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

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

where

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

For a proof of this see Appendix B.

The quantity α which now appears in the updating formulas 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

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

where J_k is an $n \times n$ nonsingular matrix. This is the representation used by Khachiyan [1979]. Observe that $\|z\| \leq 1$ is equivalent to $\|J_k^{-1}(x - x_k)\| \leq 1$; hence it follows from (2.4) and (4.3) that $B_k = J_k J_k^T$. In terms of the z variables, E_k is the unit ball and H is the half-space $\hat{H} = \{z \in \mathbb{R}^n | \hat{a}^T z \leq -\alpha\}$, where $\hat{a} = J_k^T a/\|J_k^T a\|$. Consequently, in this transformed space $|\alpha|$ is the ordinary Euclidean distance of the bounding hyperplane of $\hat{H}$ from the origin. If $\alpha > 0$ then, clearly, the origin lies outside of $\hat{H}$, or in the untransformed space, x_k lies outside of H. If $\alpha \leq -1$ E_k is contained in H; if $-1 < \alpha \leq 1$ the set $E_k \cap H$ is nonempty; and if $\alpha > 1$ the set $E_k \cap H$ is empty, implying that (2.1) is infeasible. The same statement holds if E_k and H are replaced by the unit ball $\{z \in \mathbb{R}^n | \|z\| \leq 1\}$ and $\hat{H}$, respectively.

Formulas (2.5), (2.6), (4.1) and (4.2) are only valid for determining E_{k+1} for $-1/n \leq \alpha \leq 1$; if $\alpha < -1/n$ the smallest ellipsoid containing $E_k \cap H$ is E_k. Moreover, for $-1/n \leq \alpha \leq 1$ the ratio of the volume of E_{k+1} to that of E_k is

$$r(\alpha) = ((n^2(1 - \alpha^2))/(n^2 - 1))^{(n-1)/2}(n(1 - \alpha)/(n + 1)). \tag{4.4}$$

This ratio decreases monotonically from one, when $\alpha = -1/n$ and $E_{k+1} = E_k$, to zero, when $\alpha = 1$ and E_{k+1} degenerates to a point. For $-1/n \leq \alpha < 0$ the cut $a^T x \leq \beta$ is "shallow," i.e. $E_k \cap H$ contains more than one-half of E_k including x_k. Clearly such shallow cuts need never be used.

By computing α for each inequality in (2.1) we can select the deepest cut possible; that is, the cut corresponding to the largest α. If this or any other α is greater than one, then the system (2.1) is infeasible. Using deep cuts should help to speed up the ellipsoid algorithm. However, as we shall show in Appendix C, the improvement obtained can be rather disappointing.

Krol and Mirman, and Goffin [1979b] give relaxations of the deep-cut version of the ellipsoid method by allowing different formulas 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.

Surrogate Cuts

By combining inequalities in (2.1) one can sometimes obtain cuts that are deeper than any cut generated by a single constraint in (2.1). Any cut of the form $u^T A^T x \leq u^T b$, (i.e. $a^T x \leq \beta$, with $a = Au$ and $\beta = u^T b$) is valid as long as $u \geq 0$, for then no points that satisfy (2.1) are cut off by this inequality. In Goldfarb and Todd [1980], where the term "surrogate" cut was introduced, and in Krol and Mirman the idea of using such cuts with the ellipsoid method was proposed. Clearly the "best" or "deepest" surrogate cut is one which can be obtained by solving

$$\max_{u \geq 0} u^T (A^T x_k - b)/(u^T A^T B A u)^{1/2},$$

which is equivalent to solving a quadratic programming problem. Let

$$\bar{A}^T x \leq \bar{b} \tag{4.5}$$

be any subset of the constraints (2.1), where the columns of $\bar{A}$ are linearly independent and at least one of the constraints in this subset is violated by x_k. One can easily prove (see Goldfarb and Todd) that if

$$\bar{u} = (\bar{A}^T B_k \bar{A})^{-1} (\bar{A} x_k - \bar{b}) \tag{4.6}$$

is nonnegative, then the surrogate cut $\bar{u}^T \bar{A}^T x \leq \bar{u}^T \bar{b}$ is deepest with respect to that subset. It is shown in Goldfarb and Todd, and Todd [1979], 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 Goldfarb and Todd it is recommended that only surrogate cuts which can be generated from two constraints be considered.

Krol and Mirman 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 Krol and Mirman which is based upon these observations can be viewed as a relaxation method for solving the feasibility problem for (2.1) independent of the ellipsoid algorithm. This is illustrated in Figure 4. If implemented efficiently the main computational cost on each step comes from the computation of the inner products of all m constraint normals with the normal to the current deepest (surrogate) cut in the metric B_k. If deepest cuts are to be found efficiently, the quantities $a_j^T B_k a_j$, $j = 1, \cdots, m$ should be updated on each iteration as described in Goldfarb and Todd.

Parallel Cuts

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

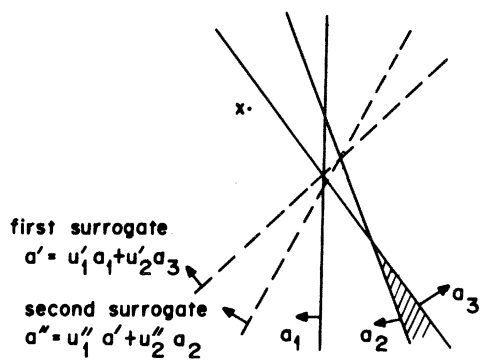

Figure 4. Krol and Mirman's iterative procedure for generating surrogate cuts.

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 $\alpha = (a^T x_k - \beta)/\sqrt{a^T B_k a}$ and $\hat{\alpha} = (\hat{\beta} - a^T x_k)/\sqrt{a^T B_k a}$, and suppose that $\alpha\hat{\alpha} < 1/n$ and $\alpha \leq -\hat{\alpha} \leq 1$. Then formulas (2.5)–(2.6) with

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

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

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

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 $\delta = (n^2/(n^2 - 1))(1 - \alpha^2)$; that is, rank (B_{k+1}) = rank $(B_k) - 1$ and E_{k+1} becomes flat in the direction of a.

As in the case of deep cuts, Shor and Gershovich were the first to suggest the use of parallel cuts and provide formulas for implementing them. They also derive formulas for circumscribing an ellipsoid (of close to minimum volume) about the region of a unit ball bounded by two or more hyperplanes which intersect in the interior of this ball and whose normals are mutually obtuse. The formulas for parallel cuts were also derived independently by König and Pallaschke, Krol and Mirman (private communication), and Todd [1980]. Proofs that they give the ellipsoid of minimum volume can be found in König and Pallaschke, and Todd [1980].

Given a violated constraint, say $a^T x \leq \beta$, König and Pallaschke show how to generate a parallel constraint $-a^T x \leq -\hat{\beta}$ which does not cut off

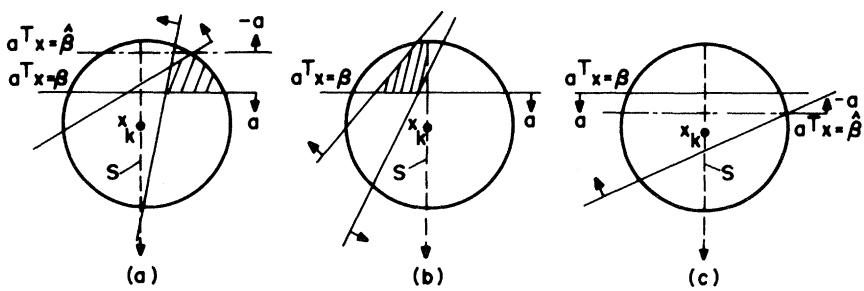

Figure 5. Generation of parallel constraint for the case $B_k = I$. (a) Parallel constraint derived from the better of two candidates. (b) No parallel constraint possible. (c) Parallel constraint indicating an empty slice.

any points in E_k that satisfy (2.1) and which yields a slice $\{x \in E_k | \hat{\beta} \leq a^T x \leq \beta\}$ of E_k that has minimum volume. Whenever there is a constraint whose (outward) normal makes an obtuse angle with a in the metric defined by B_k and whose bounding hyperplane intersects the semi-infinite open line segment $\mathscr{S} = \{x | x = x_k - \tau B_k a/\sqrt{a^T B_k a},\ \tau < 1\}$ this can be done; see Figure 5 (a) and (b). Formulas for the appropriate $\hat{\beta}$ and corresponding $\hat{\alpha}$ are given in König and Pallaschke. Note that the case $\hat{\beta} > \beta$ can occur, as illustrated in Figure 5 (c), indicating that the set $\{x \in E_k | a^T x \leq \beta\}$ is empty.

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

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

where A^T is $m \times n$. There are several ways this problem can be attacked by the ellipsoid algorithm. We discuss each approach both from a theoretical viewpoint, to establish the existence of a polynomial algorithm for (5.1), and from a practical viewpoint. We conclude this section with a discussion of how to compute in polynomial time an exact optimal solution from a sufficiently accurate approximation.

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

The problem dual to (5.1) is

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

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) respectively if and only if they solve the system of linear inequalities

$$\begin{aligned} A^T x &\leq b \\ -x &\leq 0 \\ -Ay &\leq -c \\ -y &\leq 0 \\ -c^T x + b^T y &\leq 0. \end{aligned} \tag{5.3}$$

Hence we may apply the ellipsoid method to (5.3) to solve (5.1) and (5.2) as pointed out by Gács and Lovász. Clearly, a polynomial algorithm for linear inequalities yields in this way a polynomial algorithm for linear programming problems. In addition this method directly produces an optimal dual solution.

From a practical viewpoint there are several disadvantages to this approach. The ellipsoid algorithm is applied to a system of linear inequalities in $\mathbb{R}^{m+n}$; the high dimensionality slows convergence. In many practical problems the feasible region of (5.1) is bounded and explicit bounds are known; thus a method working only in primal space can be initialized with an ellipsoid of large but not astronomical volume, speeding convergence. Bounds on the dual variables may be harder to obtain and hence the initial ellipsoid for (5.3) may be much larger than necessary. Next, all solutions to (5.3) lie in the hyperplane $c^T x = b^T y$; hence, even if (5.3) is feasible, the feasible set has zero volume. Thus perturbation of the right hand sides in (5.3) (or a related modification of the algorithm) is necessary. Moreover, even if (5.3) is feasible, the volume of the feasible set of the perturbed problem will be very small; thus the number of

iterations is likely to be very large. Another disadvantage is that if the algorithm determines that (5.3) is infeasible, yet no feasible primal or dual solution is produced, it is not clear whether (5.1) is infeasible or unbounded.

Some of these difficulties can be mitigated by the strategy used to choose constraints of (5.3) to generate cuts for the ellipsoid algorithm. Note first that, except for its final constraint, (5.3) separates into two subsystems. Thus if no cut is based on this last constraint, the matrix B_k defining the ellipsoid E_k remains block diagonal, with two blocks corresponding to the x- and the y-variables. When such a matrix B_k is updated, as long as the final constraint is not used as the cut, only one of the blocks is (nontrivially) updated, and only one of x_k and y_k changes. Thus the high dimensionality is not too drastic a problem until feasibility is reached. It seems reasonable to base cuts only on the primal constraints until a primal feasible x is generated, then only on the dual constraints until a dual feasible y is generated. (A similar strategy can be used for problems with block angular structure, i.e. block diagonal constraints with coupling constraints; note that staircase systems can be permuted to this form. In this case, B_k remains block diagonal until a cut based on a coupling constraint is used—such cuts should be postponed as long as possible.) Suppose that, before perturbation of (5.3) as in Section 2, L bits are necessary to define the system. Then one can show that, using the strategy above, if no primal feasible solution (to the perturbed system) is generated in $6n(m + n + 1)L$ steps, (5.1) is infeasible; if a primal feasible solution is generated in k steps but then no dual feasible solution is generated after a further min $\{k + 6m(m + n + 1)L, 6(m + n)(m + n + 1)L - k\}$ steps, (5.1) is unbounded.

Jones and Marwil [1980a] present a variant of this approach of simultaneously solving the primal and dual using the complementary slackness conditions for (5.1) and (5.2) to reduce the dimensionality of the problem as iterations are performed. If α, given by (4.2), is less than -1 for one of the constraints of (5.1) or (5.2), then the ellipsoid E_k is contained in the interior of the halfspace associated with that constraint; hence, one can conclude that the complementary constraint must be binding at any solution to (5.3). Consequently, when such a situation occurs, Jones and Marwil project the current iterate x_k onto this binding constraint and collapse the ellipsoid into its intersection with that constraint. All constraints with $\alpha < -1$ can be temporarily eliminated. When a solution to (5.3) excluding these constraints is found, if they are satisfied by that solution we are done. Otherwise it is necessary to continue iterations after reintroducing any violated constraints into the problem.

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

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

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

Bisection Method

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 current ellipsoid E_k given by (2.4) then $\bar{\zeta} = c^T x_k + (c^T B_k c)^{1/2}$ 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 approach has the advantage of operating only in $\mathbb{R}^n$ (except for possibly one application in $\mathbb{R}^m$). Polynomial-time algorithms combining bisection with the ellipsoid method are given by Padberg and Rao [1980a] for linear programming, and by Kozlov et al. [1979] for convex quadratic programming.

From a practical viewpoint, the main disadvantage of bisection 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

We start as before by generating a feasible solution $\underline{x}$ to (5.1). We next

consider (5.4) with $\zeta \leftarrow \underline{\zeta} = c^T\underline{x}$. Although $\underline{x}$ is feasible in this problem, we may proceed with the ellipsoid algorithm using the cut $c^Tx \geq c^T\underline{x} = \zeta$, since the next ellipsoid can be defined even when the current iterate lies on the chosen cut. Hence in this method, we are always considering feasible systems (except possibly the first). Whenever a feasible iterate x_k satisfies $c^Tx_k > c^T\underline{x} = \underline{\zeta}$, we set $\underline{x} \leftarrow x_k$ and $\zeta \leftarrow \underline{\zeta} \leftarrow c^Tx_k$ and proceed as above.

This method is probably the most efficient for practical implementation. It always considers feasible systems and never backtracks. All computation takes place in $\mathbb{R}^n$. If the feasible region of (5.1) is bounded and known to lie in E_0, then upper bounds $\bar{\zeta}_k$ can be found;

$$\bar{\zeta}_k = \min\{\bar{\zeta}_{k-1},\ c^Tx_k + (c^TB_kc)^{1/2}\}.$$

Computation can be terminated when $\underline{\zeta}$ and $\bar{\zeta}_k$ are sufficiently close. A refinement to the method to improve its performance is to set $\underline{x} \leftarrow x_k + \theta s$ whenever x_k is feasible; here s is an ascent direction (e.g., $s = c$ or $s = B_kc$) and θ is as large as possible so that $\underline{x}$ is feasible.

We mention briefly the modifications required to obtain a polynomial algorithm using this strategy. First we use the ellipsoid algorithm to determine whether (5.1) is feasible. If so, we next determine whether it is unbounded, by applying the ellipsoid algorithm either to the constraints of (5.2) or to the system

$$A^Tx \leq 0$$
$$-x \leq 0$$
$$-c^Tx \leq -1;$$

any solution to these inequalities yields an unbounded ray for (5.1). Suppose we have determined that (5.1) has a finite optimal solution. Let this, as yet unknown, solution be x^* and have value ζ^*. Let us assume that the feasible region of (5.1) contains a ball $S(a_0, r)$; otherwise we ensure this by making suitable perturbations. Then the feasible region also contains the "cone" C with vertex x^* and this ball as its base. For any $\zeta < \zeta^*$, we can easily obtain a large enough lower bound on the volume of $\bar{C} = C \cap \{x \in \mathbb{R}^n \mid c^Tx \geq \zeta\}$ so that the "sliding objective function" method for (5.1) obtains feasible solutions with objective function values within ϵ of ζ^* in a number of steps polynomial in L, $|\log 1/r|$ and $\log(1/\epsilon)$. Note that $\bar{C}$ assumes the role played by the ball $S(a_0, r)$ in the inequality case. From these solutions optimal solutions can be obtained when ϵ is sufficiently small.

The sliding objective function method was first proposed by Iudin and Nemirovskii [1976b] and Shor [1977a]. Grötschel et al. use it as a tool for demonstrating polynomial equivalence of certain combinatorial optimization problems. Goldfarb and Todd add the refinement of stepping as

far as possible in a steepest ascent or Newton-like direction, while still remaining feasible, before effecting an objective function cut. Pickel [1979] proposes stepping to a vertex by a simplex-like approach before effecting an objective function cut.

Exact Solutions from Approximate Solutions

Thus far we have only addressed the approximate solution of linear programming problems. For $\epsilon > 0$, x is defined to be an *ϵ-approximate solution* of (5.1) if there exist a feasible solution y and an optimal solution x^* of (5.1) such that $\| y - x \| \leq \epsilon$ and $c^T(x^* - x) \leq \epsilon$. A nice technique for obtaining an exact solution from an ϵ-approximate solution in polynomial time is discussed in Grötschel et al. It involves choosing an appropriately small $\epsilon > 0$ and rounding an ϵ-approximate solution, as we shall explain. We will assume that (5.1) is known to have an optimal solution—we have already observed that primal and dual feasibility can be checked in polynomial time.

Suppose that Δ is a positive integer and x^* is a rational vector of the form

$$(p_1/q_1, \cdots, p_n/q_n)^T; \qquad p_i, q_i \text{ integer and } |q_i| \leq \Delta, \quad i = 1, \cdots, n. \tag{5.5}$$

Given $x \in \mathbb{R}^n$ such that x^* is in the interior of the ball $S(x, 1/(2\Delta^2))$, then x^* is the unique rational vector of form (5.5) in this ball. For $y \in S(x, 1/(2\Delta^2))$ implies that $\| y - x^* \| \geq 1/\Delta^2$, but if y is also of the form (5.5) and for some j, $y_j \neq x_j^*$, then $| y_j - x_j^* | \geq 1/\Delta^2$ implying that $\| y - x^* \| \geq 1/\Delta^2$. 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 p/q having $|q| \leq \Delta$ by the method of continued fractions (see Niven and Zuckerman [1966]).

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 et al. point out that one can replace the objective function vector c of (5.1) by a perturbed vector $d = \gamma^n c + (\gamma^0, \cdots, \gamma^{n-1})^T$ such that the problem

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

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 $\epsilon > 0$

$$\|x^* - x\| < 1/(2\Delta^2) \tag{5.7}$$

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

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

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 Grötschel et al. 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 $1/\epsilon = n^{3/2}\, 2^{(n^{2+2n+2})L+2n+5} + \|c\| n^{1/2}\, 2^{2nL+3L+5}$. 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.

Continued fractions are also used by Kozlov et al. to round an approximate optimal objective function value to the exact optimal value in polynomial time. (See Section 7.)

6. IMPLEMENTATION

In our description of the ellipsoid algorithm, we have followed Gács and Lovász 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 formulas for determining, x_{k+1} and B_{k+1}, and hence E_{k+1}. Unfortunately, however, if these formulas 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 used is to represent the ellipsoid E_k as in (4.3)—i.e. by its center x_k and a nonsingular matrix J_k which transforms the unit ball into E_k shifted to the origin. Recall that

$$B_k = J_k J_k^T. \tag{6.1}$$

If we make this substitution in formulas (2.5)–(2.6) and define $\hat{a}_k = J_k^T a / \|J_k^T a\|$ and $w_k = J_k \hat{a}_k$, we obtain

$$x_{k+1} = x_k - \tau w_k \tag{6.2}$$

and

$$\begin{aligned} B_{k+1} &= \delta(J_k J_k^T - \sigma J_k \hat{a}_k \hat{a}_k^T J_k^T) \\ &= \delta J_k (I - \sigma \hat{a}_k \hat{a}_k^T) J_k^T \\ &= \delta J_k (I - \pi \hat{a}_k \hat{a}_k^T)(I - \pi \hat{a}_k \hat{a}_k^T) J_k^T \end{aligned}$$

where $1 - \pi = \pm(1 - \sigma)^{1/2}$. From this we obtain that

$$J_{k+1} = \delta^{1/2} J_k (I - \pi \hat{a}_k \hat{a}_k^T). \tag{6.3}$$

It is evident from (6.1) that J_k can be replaced by $J_k Q_k$, where Q_k is an $n \times n$ orthogonal matrix, (i.e. $Q_k^T Q_k = I$) without any effect on the ellipsoid E_k. Consequently, although B_k in (2.4) is unique, J_k in (4.3) is not. This also follows from the observation that replacing J_k by $J_k Q_k$ in (4.3) corresponds to first rotating the ball $\{z \in \mathbb{R}^n \mid \|z\| \leq 1\}$ before applying J_k to it. If one chooses Q_{k+1} to be an orthogonal matrix whose first column is the vector $\hat{a}_k$, one obtains

$$\begin{aligned} J'_{k+1} &= J_{k+1} Q_{k+1} \qquad (6.4) \\ &= \delta^{1/2} J_k (I - \pi \hat{a}_k \hat{a}_k^T) Q_{k+1} \\ &= J_k \delta^{1/2} (Q_{k+1} - \pi \hat{a}_k e_1^T) = J_k Q_{k+1} A_k \qquad (6.5) \end{aligned}$$

where e_1 denotes the first column of I and $A_k = \operatorname{diag}(\delta^{1/2}(1 - \pi), \delta^{1/2}, \cdots, \delta^{1/2})$. Except for a factor of $2^{1/8n^2}$ introduced by Khachiyan to compensate for roundoff effects, (6.5) is the updating formula given in Khachiyan [1979] along with (6.2).

Since B_k is symmetric positive definite, we can always choose J_k to be a lower triangular marix—i.e. $B_k = J_k J_k^T$ is the Cholesky factorization of B_k. By properly choosing Q_{k+1} in (6.4) one can ensure that J'_{k+1} is also lower triangular. This approach has the advantage of saving approximately one-half of the memory locations and one-third of the operations required for storing and updating a nontriangular J_k.

Even better yet, one can work with the factorization

$$B_k = L_k D_k L_k^T, \tag{6.6}$$

where L_k is a unit lower triangular matrix and D_k is a (positive definite) diagonal matrix. Formula (2.6) involves only a rank-one change to B_k. Hence x_k, L_k and D_k can be updated in $2n^2 + O(n)$ operations. Some care has to be exercised in how this is done since we are subtracting rather than adding a rank-one term to B_k, and roundoff errors may cause diagonal elements of D_k to vanish or become negative. A specific numerically stable algorithm which ensures that D_{k+1} is positive definite is given in Gill et al. [1975]. On the other hand, we note that 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 (see Bartels [1971]).

7. RELATIONS TO THE SIMPLEX METHOD AND OTHER METHODS

In this section we indicate some of the relations of the ellipsoid method

to the simplex method and other computational procedures. Extensions to optimization problems other than linear programming are also cited.

Some interesting observations are contained in the paper by Halfin. His view of the ellipsoid algorithm is similar to the analyses originally taken by both Shor [1970a], Iudin and Nemirovskii [1979b] and more recently by Krol and Mirman, and Jones and Marwil [1979]. In effect, all of these authors consider the space $\mathbb{R}^n$ to be transformed at each iteration so that the current ellipsoid is a sphere. Recall from (4.3) that the ellipsoid $E_k \subset \mathbb{R}^n$ is transformed by $z = T(x) \equiv J_k^{-1}(x - x_k)$ into the unit sphere. If one performs this "space dilation," to use the terminology of Shor, then instead of the ellipsoid shrinking in volume on each step, the feasible set (2.1) in the "dilated" space of the z variables, $P_k = \{z \mid (A^k)^T z \leq b^k\}$, where $A^k = J_k{}^T A$ and $b^k = b - A^T x_k$, expands. If the feasible set has a nonempty intersection with the initial ellipsoid, a unit ball, then, since this set expands and is contained in the unit ball after each transformation, after a polynomially bounded number of steps $0 \in P_k$, indicating feasibility.

One can of course view simplex pivoting in a similar way. If one uses it to find a feasible point satisfying the constraints $Ax = b$, $x \geq 0$, given in standard form, then each simplex pivot (on an infeasible row) can be thought of as an affine transformation from one space of nonbasic variables to another obtained by exchanging one nonbasic and basic variable. Partitioning A and x into basic and nonbasic parts $A = [B \mid N]$ and $x^T = [x_B{}^T, x_N{}^T]$, it is clear that simplex steps are performed until $x_N = 0$ is contained in the transformed polytope $P = \{x_N \mid x_N \geq 0, B^{-1}Nx_N \leq B^{-1}b\}$, or infeasibility is detected.

Unlike the other authors who keep an explicit form of the operator J_k and update it as in (6.3) on each iteration, Halfin applies the "space dilation operator" $\delta^{1/2}[I - \pi a_p^{(k)} a_p^{(k)T}]$ (cf. (6.3)) to A^k to obtain A^{k+1}, while simultaneously updating b^k to b^{k+1}. Here $a_p^{(k)}$ is a column of A^k corresponding to a violated constraint normalized by its length. Computationally there is little to recommend this approach to that of updating J_k, just as in the simplex method there is little to recommend the standard tableau version over the revised simplex method using an explicit inverse. In the second variant of the ellipsoid method given by Krol and Mirman, 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.*, $a_j^{(k+1)} = \delta^{1/2}(a_j^{(k)} - (\pi a_p^{(k)T} a_j^{(k)} / a_p^{(k)T} a_p^{(k)}) a_p^{(k)})$, $j = 1, \cdots, m$—and a simplex pivot. Whether this observation will lead to further 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 Halfin,

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 Goldfarb and Todd 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 full-rank system of linear equations Goffin [1979b] 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 $(1-\sigma)^k$, when the cuts are chosen cyclically. Thus in the metric corresponding to B_k the constraint normals progressively become more and more orthogonal. Goffin also shows that in this special case not only does the volume of the ellipsoid shrink, but each of its axes shrinks as a geometric series, and the iterates x_k converge to the solution $A^{-1}b$ at a geometric rate which depends only upon n.

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 (see Gács and Lovász, Padberg and Rao [1980a], Grötschel et al.); (ii) the matrix J_k which transforms a sphere into the ellipsoid E_k translated to the origin (see Shor [1970a], Khachiyan [1979], Krol and Mirman); (iii) the Cholesky or LDL^T factorization of B_k (see Jones and Marwill [1979], Goldfarb and Todd); and (iv) the problem data under the transformation induced by J_k (see Halfin, Krol and Mirman). A product form version of (iii) is discussed in Goldfarb and Todd. One of the principal computational and practical drawbacks of the ellipsoid method is that it is not possible to implement it and 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 (Goldfarb and Todd, Telgen [1980]. 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 $<(1-(1/n^2))^{n/2} < e^{-1/2n}$; hence such an algorithm is polynomial. A hybrid algorithm which combines the ellipsoid method with the simplex method is proposed in Pickel.

Although relatively little computational experience with the ellipsoid method has been reported, at present, the general consensus is that it is not a practical alternative to the simplex method for linear programming problems. A list of papers reporting computational results appears in Wolfe [1980]. In fact the only mildly encouraging results are those reported by Krol and Mirman. (However, at the Spring 1980 ORSA meeting, Krol and Mirman expressed pessimism about the practicality of the method.) Our own computational experience indicates that the slow convergence exhibited in the example analyzed in Appendix C is 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 further breakthroughs in implementation occur, it is unlikely that the ellipsoid method will replace the simplex method as the computational workhorse of linear programming.

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 stated in Section 3, the ellipsoid method was developed for solving convex (not necessarily differentiable) optimization problems (Shor [1970a], Iudin and Nemirovskii [1976b]). 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 is more likely that the ellipsoid method will be found to be of greater practical value for nonlinear and nondifferentiable problems than for linear programming. In particular, as the subgradient algorithm has been used successfully to generate bounds for certain combinatorial optimization problems (e.g. the traveling salesman problem, see Held and Karp [1970, 1971]), the ellipsoid method may be useful in this context.

Kozlov et al. [1979] 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 value, they round it to the exact optimal value $f(x^*) \equiv t/s$ in polynomial time using a continued fraction expansion. To obtain an exact optimal point for the quadratic programming problem, they first find I_m, the index set of the constraints which are active at an optimum. This is accomplished by determining the compatibility of a sequence of m systems of the form $a_i{}^T x = \beta_i$, $i \in I_k \cup \{k\}$, $a_i{}^T x \leq \beta_i$, $i = k+1, \cdots, m$ and $f(x) \leq t/s$, where $I_0 = \emptyset$ and $I_k = I_{k-1} \cup \{k\}$, $k = 1, \cdots, m$, if the kth system is compatible, and $I_k = I_{k-1}$ if it is not. They then find a point x which minimizes $f(x)$ over these active constraints. Adler et al., Chung and Murty [1979], and Jones and Marwil [1980b] have also used the ellipsoid method to attack the linear complementarity problem.

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 examine the ellipsoid method in a general framework, establish theoretical results based on the general form of the algorithm, and use those results to design polynomial-time ellipsoid algorithms for a number of combinatorial optimization problems. (Because of the history of misleading and fallacious conclusions concerning the ellipsoid method and its relationship to combinatorial problems, we hasten to add that the combinatorial problems solved in Grötschel et al.

are not among the $\mathcal{NP}$-hard problems, unless $\mathcal{P} = \mathcal{NP}$.) In this section we will attempt to portray the interesting way in which Grötschel et al. have applied the ellipsoid method to combinatorial optimization problems. (We recently learned of work by Padberg and Rao [1980c] which deals with the same general approach as Grötschel et al.)

Consider the problem

$$\text{maximize } c^T x, \qquad x \in P \subseteq R^n.$$

The basic operations of the ellipsoid method can be divided between two routines. The master, or optimization routine (Opt) performs the calculations associated with updating x_k and B_k, and testing for termination. It then calls (with $z = x_k$) a separation routine (Sep) which solves the *separation problem*:

$$\begin{aligned} &\text{given } z \in \mathbb{R}^n \text{ determine that } z \in P, \text{ or find a hyperplane} \\ &\text{that separates } z \text{ from } P, \text{ i.e. find a vector } \pi \in \mathbb{R}^n \qquad (8.1) \\ &\text{such that } \pi^T z > \pi^T y, \qquad \forall y \in P. \end{aligned}$$

The separation routine supplies the optimization routine either with the information that $x_k \in P$, or with π as in (8.1). In the former case a, the outward normal to the next cut, is set to $-c$; in the latter case it is set to π. The optimization routine can then calculate x_{k+1} and B_{k+1}. The essential character of Opt is as described in Sections 2 and 5, although certain important technical details (e.g., the use of factorizations) can vary; Sep is less predictable. In standard implementations of the ellipsoid method for conventional linear programming problems, e.g., as described in Sections 2 and 5, Sep is provided with a list of all defining inequalities $a_i^T x \leq \beta_i$ of P, which it searches for violations at $x = x_k$. However the updates of x_k and B_k in Opt depend only on the specific inequality reported by Sep, not on the manner in which it was computed, nor on any further information concerning the feasible region P. So Sep need not work by exhaustive search; any method of solving the separation problem (8.1) will do. We will see in a moment why it might be interesting in certain kinds of problems to use a separation routine not based on exhaustive search of the defining inequalities.

An Example: Network Synthesis

The character of the combinatorial results of Grötschel et al. can best be explained with the help of an example. Although the following *network synthesis problem* (see Gomory and Hu [1961, 1962, 1964]) is not discussed by Grötschel et al., it nicely illustrates their approach in attacking combinatorial problems with the ellipsoid method, and its context is familiar to most operations researchers. The data for any instance of this

problem consist of a positive integer n and two lists of $n' = n(n-1)/2$ nonnegative integers: d_{ij} and r_{ij} for all $1 \le i < j \le n$. The 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 $N = \{1, \cdots, n\}$. 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 $j \ne i$. (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.) The network must have sufficient capacity to accommodate a flow rate of r_{ij} units between nodes i and $j \ne i$ 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 $d^T x = \sum_{1 \le i < j < n} d_{ij} x_{ij}$. Note that the decision variables are permitted to assume noninteger values. For example when $n = 3$ and $d = r = (1, 1, 1)^T$, the unique optimal solution is $x = (\frac{1}{2}, \frac{1}{2}, \frac{1}{2})^T$.

In the special case where all $d_{ij} = 1$ (or d is constant over all links) Gomory and Hu [1961] (see Ford and Fulkerson [1962]) give a beautifully simple procedure for solving the synthesis problem. Gomory and Hu [1962] also point out that the general problem, although not solvable by their simple procedure, is at least a linear programming problem, one which unfortunately has an enormous number of defining inequalities. From the Max-Flow Min-Cut Theorem of Ford and Fulkerson 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 $i \in Y$, $j \in \bar{Y} \equiv N \setminus Y$

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

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

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

$$\text{subject to} \quad x(Y, \bar{Y}) \ge r_Y \quad \text{for all} \quad \emptyset \ne Y \subsetneq N, \tag{8.3b}$$

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

where $r_Y = \max\{r_{ij}: i \in Y, j \in \bar{Y}\}$. This still leaves us with $2^{n-1} - 1$ distinct inequalities of the form (8.3b), each involving only $n' = n(n-1)/2$ 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

$$l = \lfloor \log n \rfloor + \sum_{1 \le i < j \le n, r_{ij} \ne 0} \lfloor \log r_{ij} \rfloor + \sum_{1 \le i < j \le n, d_{ij} \ne 0} \lfloor \log d_{ij} \rfloor + 2(n^2 - n + 1)$$

of the problem encoding.

The first difficulty concerns the performance of the optimization routine, Opt. The size L of an explicit encoding of our linear programming problem (8.3) is not bounded by a polynomial function of l. If the parameters that prescribe the number of iterations, the number of digits of accuracy, etc., are set at values based on L, as in Sections 2 and 5, then the amount of work performed by Opt will be at least proportional to L, which can be larger than 2^{n-1}. As noted in Section 2, we can help ourselves here if we can provide $x^0 \in P$, $R > 0$, $\rho > 0$ such that $S(x^0, \rho) \subseteq P \subseteq S(x^0, R)$, and $\log R$ and $\log(1/\rho)$ are bounded by polynomials in l.

Note that (8.3) is obviously feasible—we can set $x \equiv r$, or for an initial interior point of P we can set $x_{ij}^0 = r_{ij} + 1$ for all $1 \le i < j \le n$. Now let $r_m = \max\{r_{ij}: 1 \le i < j \le n\}$ and let $R = \sqrt{n'}\,(r_m + 2)$. The ball $S(x^0, R)$ of radius R about x^0 is an appropriate initial ellipsoid. Any $x \in \mathbb{R}^{n'}$ not in $S(x^0, R)$ has at least one component less than zero or strictly larger than r_m; in the former case $x \notin P$, in the latter either $x \notin P$ or there exists $\hat{x} \in P \cap S(x^0, R)$ such that $d^T\hat{x} \le d^T x$ for all nonnegative d. (It is easy to see from the nonnegativity of all constraint coefficients in (8.3) that $S(x^0, R)$ in fact contains all extreme points of P.) Furthermore $S(x^0, R) \cap P$ contains $S(x^0, 1)$. Because $\log R$ is bounded by a polynomial function of l, this initialization guarantees that the number of iterations to compute an ϵ-approximate solution of (8.3) and the number of digits of accuracy can be set to values bounded by polynomials in l and $\log(1/\epsilon)$.

Our second major obstacle concerns the performance of the separation routine. Given a class $\mathcal{K}$ of optimization problems in $\mathbb{R}^n$ in which the feasible region of each $K \in \mathcal{K}$ is described in the problem encoding by a list of defining inequalities, then exhaustive testing of those inequalities is a straightforward algorithm that solves the separation problem in time obviously bounded by a polynomial function of the size of the encoding. In combinatorial problems such as our network synthesis problem the description of the feasible region in the problem encoding is not in terms of the explicit list of defining inequalities (8.3), and, as we have seen, enumeration of such a list cannot be performed in time bounded by a polynomial function of l, the size of the encoding. Therefore, we must provide an efficient subroutine for the associated separation problem that

somehow generates a violated inequality if the current test point is not feasible. This is especially easy for the network synthesis problem; in fact the separation routine that we will now describe was suggested by Gomory and Hu [1962] for another purpose, as we shall see later.

Suppose that $\hat{x} \in \mathbb{R}^{n'}$, and we wish to solve the separation problem for (8.3) at $\hat{x}$. Clearly if any component $\hat{x}_{ij} < 0$, then the nonnegativity constraint $x_{ij} \geq 0$ separates $\hat{x}$ from P. Suppose $\hat{x} \geq 0$. Let us solve for each $1 \leq i < j \leq n$ the $i - j$ maximum flow problem in the complete graph on N with capacity function $\hat{x}$; let v_{ij} be the computed $i - j$ maximum flow value. If $v_{ij} \geq r_{ij}$ for all $1 \leq i < j \leq n$, then $\hat{x} \in P$. If some $v_{ij} < r_{ij}$, then the flow algorithm produces an $i - j$ cutset $(Y, \bar{Y})$ having $\hat{x}(Y, \bar{Y}) = v_{ij} < r_{ij}$, so $x(Y, \bar{Y}) \geq r_{ij}$ is a violated inequality. Thus we can solve the separation problem for (8.3) in at most $n' = n(n-1)/2$ repetitions of a maximum flow routine. (In fact Gomory and Hu [1961] have shown that we will need only $n - 1$ repetitions, since one can quickly determine $n - 1$ "dominant requirements" whose satisfaction implies satisfaction of all n' requirements.) There are several polynomial-time implementations of the maximum flow algorithm; the flow algorithm of Malhotra et al. [1978] will solve each of our flow problems in $O(n^3)$ computations, each involving numbers with at most $\phi \leq \log[(2R + 2)n] + p$ 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 $\log(1/\epsilon)$. 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 $\log(1/\epsilon)$ 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 $\det A \leq [n(n-1)/2]! < (n^2)! < n^{2n^2}$, permitting us to select $\Delta = n^{2n^2}$, which allows ϵ to be chosen so that $\log(1/\epsilon)$ is polynomial in l. (One suitable choice is $1/\epsilon = 2^{n+4} n^{2n^4+4n^3+4n^2+n} + 2^5 \|d\| n^{4n^3+6n^2+1/2}$.) Thus we achieve a polynomial-time algorithm for the network synthesis problem.

This approach may seem a roundabout attack on a straightforward problem. One could have deduced immediately that the network synthesis problem is solvable in polynomial time, since it can be posed as a linear programming problem whose size is a polynomial function of l, as noted in Gomory and Hu [1964]. First form the directed graph $G = (N, E)$, where E is the set of all n' ordered pairs (i, j), $1 \leq i < j \leq n$. Now

for each of the n' possible source-sink pairs s, t define n' flow variables $f_{i,j}^{s,t}$, one for each $(i, j) \in E$. The network synthesis problem can be posed as

$$\begin{aligned}
&\text{minimize } d^T x \\
&\text{subject to: for every } s, t \in N, \qquad s \neq t \\
&\textstyle\sum_{i+1 \le j \le n} f_{i,j}^{s,t} - \sum_{1 \le j \le i-1} f_{j,i}^{s,t} = 0 \quad \text{for all} \quad i \in N \setminus \{s, t\} \qquad (8.4) \\
&\textstyle\sum_{s+1 \le j \le n} f_{s,j}^{s,t} - \sum_{1 \le j \le s-1} f_{j,s}^{s,t} \ge r_{st} \\
&-x_{ij} \le f_{i,j}^{s,t} \le x_{ij} \quad \text{for all} \quad 1 \le i < j \le n.
\end{aligned}$$

The size of the linear programming formulation (8.4) is evidently polynomial in l, so direct application of the ellipsoid method to (8.4) yields a polynomial-time algorithm. However, while (8.3) has approximately 2^{n-1} rows and n^2 columns, the numbers of rows and of columns in (8.4) are both $O(n^4)$. (Gomory and Hu's [1961] result on dominant requirements permits this to be reduced to $O(n^3)$.) The significance of the development above is that while the total size of the formulation (8.3) is much larger than the total size of (8.4) for large n, the smaller number of columns in (8.3) yields a much better bound on the number of ellipsoid computations. In particular it illustrates that the ability to generate violated constraints efficiently can result in polynomial behavior, even if the total number of defining constraints is exponential in the size of the problem description.

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 an explicit description of the defining inequalities). In many of these problems, including many $\mathcal{NP}$-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 Garey and Johnson [1979]). To solve such problems in polynomial time it suffices to give a polynomial-time separation routine. Grötschel et al. have indicated how to accomplish this for a variety of problems including: optimum branching, undirected Chinese Postman tour, minimum weight perfect matching, maximum weight matching, minimization of a submodular set function, and stability number in a perfect graph. For the latter two problems no polynomial algorithm was previously known. Though the others were known to be in $\mathcal{P}$, the approach of Grötschel et al. is new, and the ease with which it embraces a variety of problems is provocative.

The casual reader should resist any temptation to conclude that one should be able to immediately show that $\mathcal{P} = \mathcal{NP}$ by exhibiting a polynomial-time separation routine for, say, the traveling salesman problem. Certainly one can use the approach of Grötschel et al. to establish that separation problems associated with some $\mathcal{NP}$-hard problems are also $\mathcal{NP}$-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 $\mathcal{NP}$-complete.

The forthcoming example of an $\mathcal{NP}$-hard separation problem is a special case of the problem of finding an optimum extreme point of a (possibly unbounded) polyhedron.

Optimal Extreme Points of Polyhedra

First consider the problem

$$\text{maximize } \{c^T x: x \in \text{Ext}(P)\}, \tag{8.5}$$

where $P \subseteq \mathbb{R}^n$ is a polytope (*bounded* polyhedron) given by a list of defining inequalities, $c \in \mathbb{R}^n$ is given, and $\text{Ext}(P)$ is the set of extreme points of P. The ellipsoid method, since it is a polynomial-time algorithm for linear programming, provides a polynomial-time algorithm for (8.5). However if we remove the condition that P be bounded, problem (8.5) is $\mathcal{NP}$-complete. If P is unbounded, we cannot simply solve (8.5) by employing the ellipsoid method (or the simplex method) on the associated linear programming problem

$$\text{maximize } \{c^T x: x \in P\}. \tag{8.6}$$

The difficulty is that (8.6) may have no (finite) optimal solution even when (8.5) does—i.e., $c^T x$ may assume arbitrarily large values over $x \in P$—as occurs in the example of Figure 6.

To see that the general (i.e., possibly unbounded) form of (8.5) is $\mathcal{NP}$-hard, consider the special case of a directed graph $G = (V, E)$ with distinct vertices s and t, and let P_G be the polyhedron given by the solutions $x: \mathrm{E} \to \mathbb{R}$ of

$$\sum_{(i,j)\in E} x(i, j) - \sum_{(h,i)\in E} x(h, i) = \begin{cases} 1, & i = s \\ -1, & i = t \\ 0, & i \in V \setminus \{s, t\} \end{cases}$$

$$x(i, j) \geq 0 \ \forall \ (i, j) \in E.$$

The extreme points of the unbounded polyhedron P_G are precisely the incidence vectors of directed $s - t$ paths in G. Thus the unbounded version of (8.5) includes as a special case the problem of finding a longest

directed $s - t$ path in a directed graph G. The latter problem is $\mathcal{NP}$-complete; it includes as a special case the Hamiltonian path problem and is, therefore, very closely related to the traveling salesman problem.

Example: An $\mathcal{NP}$-Hard Flow Decomposition Problem

To illustrate how the ideas of Grötschel et al. can be used to show that certain combinatorial problems are $\mathcal{NP}$-hard, we consider application of the ellipsoid method to the linear programming problem

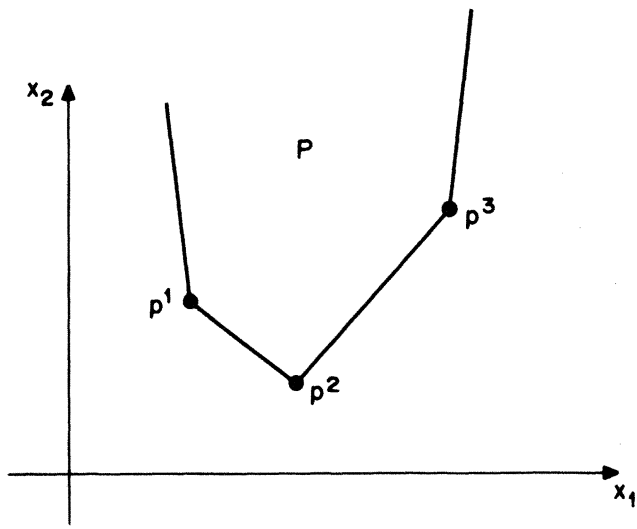

Figure 6. The two-dimensional unbounded polyhedron P has three extreme points; $\text{Ext}(P) = \{p^1, p^2, p^3\}$. With respect to the maximization of x_2 both p^1 and p^3 are locally optimal among the points in $\text{Ext}(P)$. Should the locally optimal solution p^1 be discovered by the ellipsoid method, or the simplex method, that it is globally suboptimal would not be deducible without backtracking.

$$\text{maximize } \{c^T x: x \in \hat{P}_G\}, \tag{8.7}$$

where $\hat{P}_G$ is the convex hull of the extreme points of P_G, and the problem description provides only the facets of P_G, not those of P_G. This is clearly equivalent to the longest path problem, which is $\mathcal{NP}$-complete. Yet there is no difficulty in overcoming the technical considerations needed to ensure that the computations performed by the optimization routine will be polynomial-bounded. The inherent difficulty of this problem must therefore be a reflection of an inherently difficult separation problem.

The separation problem for (8.7) concerns flow decompositions in two-terminal directed networks. Given a vector x: $E \to \mathbb{R}$ one must decide

whether $x \in \hat{P}_G$. It is easy to determine if $x \in P_G$; one simply checks $x \geq 0$ and the conservation of flow conditions. This is equivalent to determining whether x decomposes into a sum of directed $s - t$ path flows and conservative flows around directed cycles. However membership in $\hat{P}_G$ is more complicated. Given that $x \in P_G$, answering if $x \in \hat{P}_G$ is equivalent to answering whether the $s - t$ flow x can be decomposed into a sum of directed $s - t$ path flows only. It follows that the problem of determining whether a nonnegative $s - t$ flow in a directed network decomposes into a sum of directed $s - t$ path flows is $\mathcal{NP}$-hard; there exists a polynomial-time algorithm for this problem only if $\mathcal{P} = \mathcal{NP}$.

Polynomial Equivalence of Optimization and Separation

Our comments thus far have not fully conveyed the strength of the results of Grötschel et al. Suppose that $\mathcal{K}$ is a class of polytopes K each with known x^0 and $0 < \rho < R$ such that

$$S(x^0, \rho) \subseteq K \subseteq S(x^0, R\}. \tag{8.8}$$

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 $\log R$, $\log(1/\rho)$, $\log(1/\epsilon)$ and L, the size of the encoding. Grötschel et al. 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 $\|z - y\| \leq \epsilon$, or give a vector $\pi \in R^n$,

$\|\pi\| > 1$, such that $\pi^T z \geq \pi^T y - \epsilon$, $\forall y \in K$.

Furthermore they establish the converse of this result, namely, if the optimization problem is weakly solvable for $\mathcal{K}$, then so is the separation problem. These results formalize a notion that underlies much previous work in combinatorial optimization: the idea that a good characterization of a class of polytopes seems to go hand in hand with a good optimization algorithm over that class. As Lovász [1980] points out, the absence until Khachiyan [1979] of a known polynomial-time linear programming algorithm was a prominent challenge to this notion. Recent work by Karp and Papadimitriou [1980] also deals with the relationship between optimization and separation in general combinatorial optimization problems, under different assumptions than Grötschel et al.

Grötschel et al. demonstrate the polynomial equivalence of weak separation and weak optimization for any class $\mathcal{K}$ of convex (not necessarily polyhedral) bodies satisfying (8.8), and they use this additional generality in solving the stability number problem for perfect graphs in polynomial time. Since Khachiyan's work was motivated by the work of

Shor [1970a], and Iudin and Nemirovskii [1976b] 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 et al. 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 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 et al. provide an alternative to the algorithms of Edmonds [1968], and of Lawler [1976] for the (2-) matroid intersection problem.

As with the application of the ellipsoid method to general linear programming, one must be careful not to confuse the lovely results of Grötschel et al. concerning theoretical efficiency with practical considerations. They 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 guarantee the polynomial bound is orders of magnitude beyond what one would be willing to maintain in practice.

A Connection with Column Generation

It is also interesting from a practical point of view to note that the general approach of Grötschel et al. has a dual relationship with a well-known technique in linear programming, that is, column generation. In the same sense that one can iterate the ellipsoid method without explicit knowledge of all of the rows of the constraint matrix, so can one iterate the simplex method without explicit knowledge of all columns. Moreover while the amount of work performed by the ellipsoid method seems much more sensitive to the number of columns than the number of rows, experience with the simplex method has been exactly the reverse. That the network synthesis problem can have an enormous number of rows, but a manageable number of columns, coupled with our ability to easily generate rows by a flow algorithm, makes the ellipsoid method seem well suited for this problem. Note that the linear programming dual of the network synthesis problem (8.3) has few rows and many columns. Fur-

thermore the same separation routine that generates rows for the ellipsoid method could be used to generate columns for the simplex method (applied to the dual of (8.3)). Indeed Gomory and Hu [1962] proposed exactly that. (Note the substantial advantages of using column generation to solve the dual of (8.3) as opposed to solving (8.4), or the $O(n^3)$ version of (8.4).) Clearly any combinatorial problem that is solvable by the ellipsoid method in conjunction with a polynomial time separation algorithm $\mathcal{S}$ can also be solved (in its dual form) by the simplex method using $\mathcal{S}$ to generate columns. And the same problem characteristics that make the ellipsoid method well suited also commend the simplex method. Of course the simplex method (in its conventional implementations) is not polynomial. But from a practical standpoint we might not expect to achieve any better performance from the ellipsoidal approach to these problems.

9. CONCLUDING REMARKS

We conclude the main body of this paper with some remarks on the potential value of the ellipsoid method, and we address several questions raised by Khachiyan's result.

From a practical point of view, analytical and computational investigations of the ellipsoid method have not been encouraging. There are two principal reasons for this. First, the rate of convergence of the ellipsoid method, even with the use of deep cuts and surrogate cuts, is rather slow, especially when compared to practical experience with the simplex method. The worst-case bound for the simplex method, in any of its several implementations, is an extremely poor indicator of the method's actual performance; in fact, practitioners have observed that the number of iterations tends to be proportional to the number of constraints, m. On the other hand, testing thus far indicates that the worst-case bound for the ellipsoid method appears to be a better measure of the dependence of its computational effort on problem size.

Second, the ellipsoid method does not seem to be able to exploit sparsity. Thus, even if the number of iterations could be reduced significantly, the ellipsoid method would still not be a practical algorithm for solving large sparse linear programming problems unless this drawback could also be overcome. The method may be of greater interest in the solution of convex, not necessarily differentiable, optimization problems.

One criticism directed against the ellipsoid method is that it does not provide optimal dual variables and it does not lend 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, this information can readily be obtained 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, 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 Lawler [1976]), 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 Grötschel et al. 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 noncombinatorial 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 method will generate further research in this area.

APPENDIX A: COMPUTATIONAL COMPLEXITY AND LINEAR PROGRAMMING

For the reader unfamiliar with computational complexity we will attempt to convey some of the background relevant to the question of the existence of a polynomial-time linear programming algorithm. The discussion will be informal; the reader is encouraged to consult Aho et al., Garey and Johnson, and Karp [1972, 1975] for rigorous treatments of computational complexity. Note that in these references the problems that go under the familiar names linear programming, traveling salesman, etc., are not the usual optimization problems from the operations research literature, but related "yes-no" *decision*, or feasibility, problems. For example, the decision problem of determining whether there exists a traveling salesman tour of length no greater than k, a constant specified in the input, replaces the usual problem of finding a minimum length traveling salesman tour. However, for the traveling salesman problem,

linear programming, and most of the problems of interest here, the optimization and decision versions are equivalent in a certain sense. It is easy to solve the optimization problem in polynomial-time, given a polynomial-time subroutine for the decision problem, and vice versa. (See the discussion of *Turing reducibility* in Garey and Johnson.) Thus the results on $\mathcal{P}$ and $\mathcal{NP}$ described in the aforementioned papers, though they explicitly deal only with decision problems, yield information about the computational difficulty of the associated optimization problems.

In order to evaluate the effectiveness of an algorithm we might examine how its running time varies with problem size. Running time can be represented by the total number of elementary arithmetic operations: comparisons, additions, multiplications, etc. The size of a problem is taken to be the length (number of symbols) of an encoding of the problem data in which integers are represented in, say, binary form. (It is always assumed that a method of encoding the problem data is part of the problem definition.) For example the $m \times n$ linear programming problem

$$\text{maximize} \quad c^T x \quad \text{subject to} \quad A^T x \leq b,\ x \geq 0 \tag{A.1}$$

can be encoded by a list of integers: n, m, and the entries a_{ij} of A, β_i of b, and c_j of c in some specified order, separated by sign bits. The binary expansion of a nonnegative integer p has $1 + \lfloor \log p \rfloor$ bits if p is positive, and 1 bit if $p = 0$, where logarithms are base 2 and $\lfloor x \rfloor$ is the greatest integer less than or equal to x. So the length of such an encoding of the linear programming problem (A.1) is

$$\begin{aligned} L = {} & (2 + \lfloor \log n \rfloor) + (2 + \lfloor \log m \rfloor) \\ & + (2mn + \textstyle\sum_{1 \leq i \leq m, 1 \leq j \leq n, a_{ij} \neq 0} \lfloor \log |a_{ij}| \rfloor) \\ & + (2m + \textstyle\sum_{1 \leq i \leq m, \beta_i \neq 0} \lfloor \log |\beta_i| \rfloor) + (2n + \textstyle\sum_{1 \leq j \leq n, c_j \neq 0} \lfloor \log |c_j| \rfloor). \end{aligned} \tag{A.2}$$

Let $\mathcal{Q}$ denote a (generic) problem, i.e. an infinite family of (specific) problem instances $Q \in \mathcal{Q}$. For example $\mathcal{Q}$ might represent the linear programming problem (A.1), and each choice of n, m, A, b, and c constitutes an instance $Q \in \mathcal{Q}$. Let $\mathcal{A}$ be an algorithm that solves $\mathcal{Q}$. For every $Q \in \mathcal{Q}$ denote by $f_{\mathcal{A}}(Q)$ the running time of $\mathcal{A}$ in solving Q, and let $|Q|$ denote the length of the encoding of Q. In order to claim that $\mathcal{A}$ is efficient we would like to be able to exhibit for each positive integer s a "guarantee"

$$f_{\mathcal{A}}(Q) \leq g(s) \quad \text{for all} \quad Q \in \mathcal{Q} \quad \text{such that} \quad |Q| \leq s \tag{A.3}$$

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 $\mathscr{A}$ as problem size increases. $\mathscr{A}$ is said to be a *polynomial-bounded* or *polynomial-time* algorithm if there exists a polynomial function $g(s)$ satisfying (A.3). Problem $\mathscr{Q}$ is called *polynomially solvable* if there exists a polynomial-time algorithm for $\mathscr{Q}$. The class of all polynomially solvable (decision) problems is denoted by $\mathscr{P}$. 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 ℓ larger than 1, since such a change increases the length of the encoding by at most a factor of $\log \ell$.

Polynomial boundedness was proposed by Edmonds [1965] and independently by Cobham [1965] as a *theoretical* 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 $\mathscr{NP}$-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 $\mathscr{NP}$. Informally we can regard $\mathscr{NP}$ 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 for every "yes" (feasible) instance, there is some sequence of branches that leads to a "yes" answer in polynomial time.) $\mathscr{NP}$ includes a large number of well-known problems; indeed it should be clear that $\mathscr{P} \subseteq \mathscr{NP}$, 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 $\mathscr{NP}$ not in $\mathscr{P}$. The question of whether $\mathscr{P} \subsetneq \mathscr{NP}$ or $\mathscr{P} = \mathscr{NP}$ is unresolved; it is often called the "biggest" open problem in theoretical computer science. Most researchers consider it very unlikely that $\mathscr{P} = \mathscr{NP}$.

Although the $\mathscr{P} \stackrel{?}{=} \mathscr{NP}$ question is unresolved, problems that must be in

$\mathcal{NP} \setminus \mathcal{P}$, if $\mathcal{P} \neq \mathcal{NP}$ have been identified. Among these are the $\mathcal{NP}$-complete problems first examined by Cook [1971]. Essentially, Cook showed how to devise from a polynomial-depth backtrack search algorithm for any problem $\mathcal{Q}$, a polynomial-time algorithm $\mathcal{A}_{\mathcal{Q}}$ that transforms instances of $\mathcal{Q}$ into equivalent instances of the satisfiability problem in the propositional calculus. Every problem $\mathcal{Q}$ in $\mathcal{NP}$ is in this way *polynomially reducible* to satisfiability, which is itself in $\mathcal{NP}$. Hence satisfiability can be regarded to be as hard as any problem in $\mathcal{NP}$; it is said to be *complete* in $\mathcal{NP}$, or $\mathcal{NP}$*-complete*. In particular a polynomial-time algorithm $\mathcal{A}$ for satisfiability would yield for each $\mathcal{Q}$ in $\mathcal{NP}$ a polynomial-time algorithm for $\mathcal{Q}$, which first performs $\mathcal{A}_{\mathcal{Q}}$ and then $\mathcal{A}$.

Now suppose that a problem $\mathcal{Q}$ is known to be $\mathcal{NP}$-complete, e.g. the satisfiability problem, and suppose further that there is a polynomial-time reduction that takes instances of $\mathcal{Q}$ to instances of problem $\mathcal{Q}'$. Then $\mathcal{Q}'$ is said to be $\mathcal{NP}$*-hard*. (The terms "$\mathcal{NP}$-hard" and "$\mathcal{NP}$-complete" are used in several different senses, corresponding to different notions of reducibility. The use of "$\mathcal{NP}$-hard" in Section 8, as in Garey and Johnson, is in the weaker sense of Turing reducibility. A problem that is $\mathcal{NP}$-hard in any of these senses has no polynomial-time algorithm unless $\mathcal{P} = \mathcal{NP}$.) If the $\mathcal{NP}$-hard problem $\mathcal{Q}'$ can be shown to be in $\mathcal{NP}$, then $\mathcal{Q}'$ is $\mathcal{NP}$-complete. Cook's Theorem thereby simplifies the requirements for demonstrating that additional problems are also $\mathcal{NP}$-complete. He showed in this way that the clique problem for graphs is $\mathcal{NP}$-complete. Karp [1972] used Cook's Theorem to show that many well-known optimization problems, including the traveling salesman problem, are $\mathcal{NP}$-complete. A host of researchers have since contributed to the list of $\mathcal{NP}$-complete problems, which includes many familiar to all operations researchers. (More than 300 of these problems are collected in Garey and Johnson.) In fact the decision versions of most of the standard (deterministic) discrete optimization problems in the operations research literature were shown to be in $\mathcal{P}$, or to be $\mathcal{NP}$-complete, some time ago. However, one problem that until recently resisted classification was linear programming, perhaps the most widely studied of all of the problems in operations research.

Although Dantzig's simplex algorithm, in its usual implementations, has been overwhelmingly successful in the solution of real-world linear programming problems, these implementations are not polynomial-bounded. Klee and Minty [1972] gave the first example of an infinite family of linear programming problems in which the number of simplex pivots (with the "approximate steepest ascent" pivoting rule) grows like 2^n, while problem size grows like a polynomial function of n. Jeroslow [1973], Avis and Chvátal [1976], and Goldfarb and Sit [1979] have shown that similar behavior can occur with other pivoting rules. Edmonds (in

an unpublished note), Zadeh [1973], and Cunningham [1979] have shown that even within the special class of network flow problems, simplex pivoting rules can exhibit such nonpolynomial behavior.

On the other hand, it was considered extremely unlikely that linear programming might be $\mathcal{NP}$-complete, because the duality theorem of linear programming would then yield an unlikely duality for all problems in $\mathcal{NP}$. Some researchers felt that linear programming must occupy a middle position in $\mathcal{NP}$, neither in $\mathcal{P}$ nor $\mathcal{NP}$-complete. Many believed linear programming to be in $\mathcal{P}$, but failed to provide a proof. Khachiyan [1979] reported how to implement the ellipsoid method to solve the linear programming problem in polynomial time, and thus settled the issue of where linear programming resides in the $\mathcal{P} - \mathcal{NP}$ hierarchy.

APPENDIX B: MINIMUM VOLUME ELLIPSOIDS

Here we show that the formulas 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, $j = 1, 2, \cdots, n$.

Hence suppose

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

and consider the general ellipsoid

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

where $B = JJ^T$.

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

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

and

$$\begin{aligned} \tau = (1 + n\alpha)/(n + 1), \qquad \sigma = 2(1 + n\alpha)/((n + 1)(1 + \alpha)) \\ \textit{and} \quad \delta = (n^2/(n^2 - 1))(1 - \alpha^2). \end{aligned} \tag{B.3}$$

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

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

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

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

$$a_{jj} = \sum_{i=1}^{j-1} l_{ji}^2 + l_{jj}^2. \tag{B.4}$$

Thus

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

Equality holds if and only if $l_{ji} = 0$ 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 $-\alpha e_1 \pm (1 - \alpha^2)^{1/2}e_i$, $i = 2, \cdots, n$. 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 $-\alpha e_1 \pm \gamma e_i$, $i = 2, \cdots, n$, is E_+ given by (B.1)-(B.3).*

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

$$\| -y_1 - y_0\| = \| (1 - \alpha)y_1 + (\alpha y_1 + y_0)\| \leq 1 \tag{B.5}$$

and

$$\| -\alpha y_1 \pm \gamma y_i - y_0\| = \| \pm \gamma y_i + (\alpha y_1 + y_0)\| \leq 1 \tag{B.6}$$

for $i = 2, \cdots, n$.

From (B.6) we deduce that

$$\beta \equiv \| \alpha y_1 + y_0\| \leq 1 \tag{B.7}$$

and from (B.7)

$$\| (1 - \alpha)y_1\| \leq 1 + \beta. \tag{B.8}$$

Since one of $\pm\gamma y_i$ makes an acute angle with $\alpha y_1 + y_0$, (B.6) gives

$$\| \gamma y_i\|^2 + \| \alpha y_1 + y_0\|^2 \leq 1$$

or

$$\| \gamma y_i\| \leq (1 - \beta^2)^{1/2}, \qquad i = 2, \cdots, n. \tag{B.9}$$

Now (B.8) and (B.9) with Hadamard's inequality yield

$$\begin{aligned} &|\det Y| \leq \Pi_{j=1}^n \| y_j\| \\ &\leq ((1 + \beta)(1 - \beta^2)^{(n-1)/2})/((1 - \alpha)(1 - \alpha^2)^{(n-1)/2}) = f(\beta)/f(-\alpha), \end{aligned} \tag{B.10}$$

where $f(\eta) \equiv (1 + \eta)(1 - \eta^2)^{(n-1)/2}$. It is easy to check that $f'(\beta) = (1 - \beta^2)^{(n-3)/2} \times (1 - n\beta)(1 + \beta)$; thus $f(\beta)$ is maximized for $0 \leq \beta \leq 1$ by $\beta = 1/n$. Hence, assuming E_+ contains the specified points, we have

$$\begin{aligned} \text{vol } E_+/\text{vol } E &= |\det J| = 1/|\det Y| \geq (f(-\alpha))/f(1/n) \\ &= (n/(n + 1))(n^2/(n^2 - 1))^{(n-1)/2}(1 - \alpha)(1 - \alpha^2)^{(n-1)/2}. \end{aligned}$$

Now equality is achieved if and only if we have equality in (B.8), (B.9) and Hadamard's inequality in (B.10), and $\beta = 1/n$. Thus $y_1, y_2, \cdots, y_n$ are mutually orthogonal, whence $B^{-1} = J^{-T}J^{-1} = Y^TY$ and B are diagonal. Furthermore

$$e_j{}^TB^{-1}e_j = y_j{}^Ty_j = \begin{cases} (n+1)^2/(n^2(1-\alpha)^2) & \text{for } j = 1 \\ (n^2-1)/(n^2(1-\alpha^2)) & \text{for } j > 1 \end{cases}$$

from equality in (B.8) and (B.9) and $\beta = 1/n$; thus B is as given in (B.2)–(B.3). Finally, equality in (B.8) implies equality in (B.5) and that y_0 is parallel to y_1 with $\|y_0\| = 1 \pm \|y_1\|$. Then (B.7) implies that the negative sign must be taken, so that

$$y_0 = ((1 - \|y_1\|)/\|y_1\|)y_1 = -\tau y_1$$

with $\tau = (1 + n\alpha)/(n+1)$. Thus $x_0 = Jy_0 = -\tau Jy_1 = -\tau e_1$ and the proof is complete.

LEMMA B.4. *If* $-1/n \le \alpha < 1$, *the ellipsoid* E_+ *given by* (B.1)–(B.3) *contains* $E \cap H$.

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

$$B^{-1} = \operatorname{diag}((n+1)^2/(n^2(1-\alpha)^2), (n^2-1)/(n^2(1-\alpha^2)), \cdots, (n^2-1)/(n^2(1-\alpha)^2)).$$

Hence

$$\begin{aligned}(x - x_0)^TB^{-1}(x - x_0) &= ((n^2-1)/(n^2(1-\alpha^2)))\|x\|^2 \\ &\quad + ((n+1)^2/(n^2(1-\alpha)^2) \\ &\quad - (n^2-1)/(n^2(1-\alpha^2)))\xi_1{}^2 \\ &\quad + 2((n+1)(1+n\alpha)/(n^2(1-\alpha)^2))\,\xi_1 \\ &\quad + (1+n\alpha)^2/(n^2(1-\alpha)^2) \\ &= ((n^2-1)/(n^2(1-\alpha^2)))\,(\|x\|^2 - 1) \\ &\quad + (2(n+1)\,(1+n\alpha)/(n^2(1 \\ &\quad - \alpha^2)\,(1-\alpha)))\xi_1{}^2 \\ &\quad + (2(n+1)\,(1+n\alpha)/(n^2(1-\alpha)^2))\,\xi_1 \\ &\quad + 2(n+1)(1+n\alpha)\alpha/(n^2(1 \\ &\quad - \alpha^2)(1-\alpha)) + 1 \\ &= ((n^2-1)/(n^2(1-\alpha^2)))(\|x\|^2 - 1) \\ &\quad + (2(n+1)\,(1+n\alpha)/(n^2(1-\alpha^2)(1 \\ &\quad - \alpha)))(\xi_1 + \alpha)(\xi_1 + 1) + 1.\end{aligned}$$

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

The uniqueness of the minimum volume ellipsoid containing $E \cap H$ is a special case of a result of K. Löwner (see Danzer et al. [1963], p. 139): every compact set in $\mathbb{R}^n$ with positive volume has a unique circumscribing ellipsoid of minimum volume. See also John [1948]. König and Pallaschke give another proof of Theorem B.1 using this uniqueness result.

A more general version of Theorem B.1 involving cuts by two parallel hyperplanes is given in Shor and Gershovich, König and Pallaschke, and Todd [1980]; according to Shor and Gershovich this result originated with Gulinski and Polyak.

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 to 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 $x \in R^n$ satisfying

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

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

Let us start with

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

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

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

$$x_1 = (-(n-1)/(n+1), 1, \cdots, 1)^T, \quad \text{and}$$

$$B_1 = \operatorname{diag}(n^2((n-1)/(n+1))^2, n^2, \cdots, n^2).$$

Note that again each α is $\pm 1/n$.

Suppose that after k iterations, our algorithm has made $i^+_{k,j}$ cuts based on $\xi_j \ge 0$ and $i^-_{k,j}$ cuts based on $\xi_j \le 0$, where $i^{\pm}_{k,j}$ are nonnegative integers with $i^-_{k,j} - i^+_{k,j} = 0$ or 1 and $\sum_{j=1}^n (i^+_{k,j} + i^-_{k,j}) = k$. Then one can show by induction that

$$\xi_{k,j} = (-(n-1)/(n+1))^{i^+_{k,j} + i^-_{k,j}}, \qquad j = 1, \cdots, n$$

and B_k is the diagonal matrix

$$B_k = n^2 \sum_{j=1}^{n} ((n-1)/(n+1))^{2(i_{k,j}^{+}+i_{k,j}^{-})} e_j e_j^T;$$

hence again each α is $\pm 1/n$.

If the algorithm always chooses the (deepest) cut with the lowest index j—i.e., it alternatively chooses $\xi_1 \leq 0$ and $-\xi_1 \leq 0$—then $\xi_{k,1} \to 0$ but $\xi_{k,2} = \cdots = \xi_{k,n} = 1$ for all k; hence the sequence $\{x_k\}$ does not converge to the feasible set. Moreover, if the same algorithm is used on the infeasible problem obtained by replacing the constraint $-\xi_n \leq 0$ by $-\xi_n \leq -\frac{1}{2}$, the same sequence of iterates $\{x_k\}$ is generated, and infeasibility is not detected, α is always less than one, no matter how many iterations are performed (assuming exact arithmetic is used). This example demonstrates the significance of perturbing the feasible set, if it is not known to have positive volume. (See Section 2.)

Choosing the coordinates in turn in the example results in a sequence of cuts compatible with perturbation. Hence the arguments of Section 2 guarantee that feasibility will be detected after at most $6n(n+1)L$ iterations. Even so, the convergence is still extremely slow being only linear with ratio $((n-1)/(n+1))^{1/n}$. This is not a great improvement over that obtained when cuts through the center are employed. With $\alpha = 1/n$, the volume reduction achieved using a deep cut is equivalent to that obtained with 4 to 5 iterations with cuts through the center; thus we might expect the total number of iterations to be reduced by a factor of four or five. Indeed, with $r(\alpha)$ denoting the volume reduction using a deep cut (see (4.4)), it can be shown using Taylor expansions of $\ln[r(1/n)/(r(0))^4]$ and $\ln[r(1/n)/(r(0))^5]$ that $(r(0))^5 \leq r(1/n) = (n-1)/(n+1) \leq (r(0))^4$ for $n \geq 2$.

ACKNOWLEDGMENT

This work was supported in part by National Science Foundation Grants ENG-7910807, MCS-8006065, and ECS-7921279, by U.S. Army Research Office Grant DAAG29-77-G-0114 and by an Alfred P. Sloan Foundation Research Fellowship awarded to the first author.

M. Akgül, A. H. G. Rinnooy Kan and A. Schrijver offered several suggestions that improved the exposition.

REFERENCES

Adler, I., 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.

AGMON, S. 1954. The Relaxation Method for Linear Inequalities. *Can. J. Math.* **6,** 382–392.

AHO, A. V., J. E. HOPCROFT AND J. D. ULLMAN. 1976. *The Design and Analysis of Computer Algorithms.* Addison-Wesley, Reading, Mass.

AVIS, D., AND V. CHVÁTAL. 1976. Notes on Bland's Pivoting Rule. *Math. Program. Stud.* **8,** 24–34.

BARTELS, R. H. 1971. A Stabilization of the Simplex Method. *Numer. Math.* **16,** 414–434.

CHUNG, S. J., AND K. G. MURTY. 1979. A Polynomially Bounded Algorithm for Positive Definite Symmetric LCPs, Technical Report No. 79-10, Department of Industrial and Operations Engineering, The University of Michigan, Ann Arbor (December).

COBHAM, A. 1965. The Intrinsic Computational Difficulty of Functions. In *Proc. 1964 International Congress for Logic, Methodology, and Philosophy of Science*, pp. 24–30, Y. Bar-Hillel (ed.). North-Holland, Amsterdam.

COOK, S. A. 1971. The Complexity of Theorem-Proving Procedures. *Proc. ACM Symp. Theory Comput.* **3,** 151–158.

CUNNINGHAM, W. H. 1979. Theoretical Properties of the Network Simplex Method. *Math. Opns. Res.* **4,** 196–208.

DANTZIG, G. B. 1963. *Linear Programming and Extensions.* Princeton University Press, Princeton, N.J.

DANZER, L., B. GRÜNBAUM AND V. KLEE. 1963. Helly's Theorem and Its Relatives. *Proc. A.M.S. Symposium on Convexity*, pp. 101–177, V. Klee (ed.). American Mathematical Society, Providence, R.I.

EDMONDS, J. 1965. Paths, Trees and Flowers. *Can. J. Math.* **17,** 449–467.

EDMONDS, J. 1968. Matroid Partition. In *Mathematics of the Decision Sciences,* Part I, pp. 335–345. G. B. Dantzig and A. F. Veinott, Jr. (eds.) Lectures in Applied Mathematics 11, Amer. Math. Soc., Providence, R.I.

ERMOLEV, IU. M. 1966. Methods of Solution of Nonlinear Extremal Problems. *Kibernetika* **2**(4), 1–17 (translated in *Cybernetics* **2**(4), 1–14, 1966).

FORD, L. R. JR., AND D. R. FULKERSON. 1962. *Flows in Networks.* Princeton University Press, Princeton, N.J.

GÁCS, P., AND L. LOVÁSZ. 1981. Khachiyan's Algorithm for Linear Programming. *Math. Program. Stud.* **14,** 61–68.

GAREY, M. R., AND D. S. JOHNSON. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman, San Francisco.

GILL, P. E., W. MURRAY AND M. A. SAUNDERS. 1975. Methods for Computing and Modifying the LDV Factors of a Matrix. *Math. Comput.* **29,** 1051–1077.

GOFFIN, J.-L. 1978. Acceleration in the Relaxation Method for Linear Inequalities and Subgradient Optimization, Working Paper 79-10, 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).

GOFFIN, J.-L. 1979a. On the Non-polynomiality of the Relaxation Method for System of Inequalities, Faculty of Management, McGill University, Montreal, Quebec (November).

GOFFIN, J.-L. 1979b. Convergence of a Cyclic Shor-Khachian Method for Systems

of Linear Equalities, Working Paper No. 79-54, Faculty of Management, McGill University, Montreal, Quebec (December).

GOLDFARB, D. AND W. Y. SIT. 1979. Worst Case Behavior of the Steepest Edge Simplex Method. *Discrete Appl. Math.* **1,** 277–285.

GOLDFARB, D., AND M. J. TODD. 1980. Modifications and Implementation of the Shor-Khachian Algorithm for Linear Programming, Technical Report 406, Department of Computer Science, Cornell University, Ithaca, N.Y. (January).

GOMORY, R. E., AND T. C. HU. 1961. Multi-Terminal Network Flows. *J. SIAM* **9,** 551–570.

GOMORY, R. E., AND T. C. HU. 1962. An Application of Generalized Linear Programming to Network Flows. *J. SIAM* **10,** 260–283.

GOMORY, R. E., AND T. C. HU. 1964. Synthesis of a Communication Network. *J. SIAM* **12,** 348–369.

GRÖTSCHEL, M., L. LOVÁSZ AND A. SCHRIJVER. 1981. The Ellipsoid Method and Its Consequences in Combinatorial Optimization. *Combinatorica* (to appear).

HALFIN, S. The Sphere Method for Khachiyan's Algorithm, undated (received March 1980), Bell Telephone Laboratories, Holmdel, N.J.

HELD, M., AND R. M. KARP. 1970. The Traveling Salesman Problem and Minimum Spanning Trees. *Opns. Res.* **18,** 1138–1162.

HELD, M., AND R. M. KARP. 1971. The Traveling Salesman Problem and Minimum Spanning Trees: Part II. *Math. Program.* **1,** 6–25.

HOFFMAN, A. J. 1952. On Approximate Solutions of Systems of Linear Inequalities. *J. Res. Natl. Bur. Stand.* **49,** 263–265.

IUDIN, D. B., AND A. S. NEMIROVSKII. 1976a. Evaluation of the Informational Complexity of Mathematical Programming Problems. *Ekonomika i Matematicheskie Metody* **12,** 128–142 (translated in *Matekon: Translations of Russian and East European Math. Economics* **13,** 3–25, Winter '76–'77).

IUDIN, D. B., AND A. S. NEMIROVSKII. 1976b. Informational Complexity and Effective Methods of Solution for Convex Extremal Problems. *Ekonomika i Matematicheskie Metody* **12,** 357–369 (translated in *Matekon: Translations of Russian and East European Math. Economics* **13,** 25–45, Spring '77).

JEROSLOW, R. 1973. The Simplex Algorithm with the Pivot Rule of Maximizing Criterion Improvement. *Discrete Math.* **4,** 367–377.

JOHN, F. 1948. Extremum Problems with Inequalities as Subsidiary Conditions. In *Courant Anniversary Volume*, pp. 187–204. Interscience, New York.

JONES, P. C., AND E. S. MARWIL. 1979. *A Variant of Khachiyan's Algorithm for Linear Programming.* EG & G Idaho, Inc., P.O.B. 1625, Idaho Falls, Idaho 83415 (November).

JONES, P. C., AND E. S. MARWIL. 1980a. *A Dimensional Reduction Variant of Khachiyan's Algorithm for Linear Programming Problems.* EG & G Idaho, Inc., P.O.B. 1625, Idaho Falls, Idaho 83415 (January).

JONES, P. C., AND E. S. MARWIL. 1980b. *Solving Linear Complementarity Problems with Khachiyan's Algorithm.* EG & G Idaho, Inc., P.O.B. 1625, Idaho Falls, Idaho 83415 (January).

KARP, R. M. 1972. Reducibility Among Combinatorial Problems. In *Complexity of Computer Computations*, R. E. Miller and J. W. Thatcher (eds.). Plenum Press, New York.

KARP, R. M. 1975. On the Computational Complexity of Combinatorial Problems. *Networks* **5,** 45–68.

KARP, R. M., AND C. H. PAPADIMITRIOU. 1980. On Linear Characterizations of Combinatorial Optimization Problems, pp. 1–9. In *Proc. 21st Annual Symposium on Foundations of Computer Science.*

KHACHIYAN, L. G. 1979. A Polynomial Algorithm in Linear Programming. *Doklady Akademiia Nauk SSSR* **244,** 1093–1096 (translated in *Soviet Mathematics Doklady* **20,** 191–194, 1979).

KHACHIYAN, L. G. 1980. Polynomial Algorithms in Linear Programming. *Zhurnal Vychisditel'noi Matematiki i Matematicheskoi Fiziki* **20,** 51–68.

KLEE, V., AND G. L. MINTY. 1972. How Good is the Simplex Algorithm? In *Inequalities III,* pp. 159–175, O. Shisha (ed.). Academic Press, New York.

KÖNIG, H., AND D. PALLASCHKE. 1981. On Khachiyan's Algorithm and Minimal Ellipsoids. *Numerische Mathematik* **38,** 211–223. University of Bonn.

KOZLOV, M. K., S. P. TARASOV AND L. G. KHACHIYAN. 1979. Polynomial Solvability of Convex Quadratic Programming. *Doklady Akademiia Nauk SSSR* **248** (translated in *Soviet Mathematics Doklady* **20,** 1108–1111, 1979).

KROL, Y., AND B. MIRMAN. Some Practical Modifications of Ellipsoid Method for LP Problems (undated, received January 1980). Arcon, Inc., Boston.

KROL, Y., AND B. MIRMAN. Private communication.

LAWLER, E. L. 1976. *Combinatorial Optimization: Networks and Matroids.* Holt, Rinehart & Winston, New York.

LAWLER, E. L. 1980. The Great Mathematical Sputnik of 1979. *The Sciences* (September).

LEMARECHAL, C. 1975. An Extension of Davidon Methods to Non-Differentiable Problems. *Math. Program. Stud.* **3,** 95–109.

LEVIN, A. IU. 1965. On an Algorithm for the Minimization of Convex Functions. *Doklady Akademiia Nauk SSSR* **160,** 1244–1247 (translated in *Soviet Mathematics Doklady* **6,** 286–290, 1965).

LOVÁSZ, L. 1980. A New Linear Programming Algorithm—Better or Worse than the Simplex Method? *Math. Intelligencer* **2,** 141–146.

MALHOTRA, V. M., M. P. KUMAR AND S. N. MAHESHWARI. 1978. An $O(V^3)$ Algorithm for Finding Maximum Flows in Networks. *Info. Proc. Letters* **7,** 277–278.

MOTZKIN, T., AND I. J. SCHOENBERG. 1954. The Relaxation Method for Linear Inequalities. *Can. J. Math.* **6,** 393–404.

NEMIROVSKII, A. S., AND D. B. IUDIN. 1977. Optimization Methods Adapting to the "Significant" Dimension of the Problem. *Automatika i Telemekhanika* **38**(4), 75–87 (translated in *Automation and Remote Control* **38**(4), 513–524, 1977).

NEWMAN, D. J. 1965. Location of the Maximum on Unimodal Surfaces. *J. Assoc. Comput. Mach.* **12,** 395–398.

NIVEN, I., AND H. S. ZUCKERMAN. 1966. *An Introduction to the Theory of Numbers.* John Wiley & Sons, New York.

PADBERG, M. W., AND M. R. RAO. 1980a. The Russian Method for Linear Inequalities, Graduate School of Business Administration, New York Univer-

sity, New York (January).

PADBERG, M. W., AND M. R. RAO. 1980b. The Russian Method for Inequalities II: Approximate Arithmetic, Graduate School of Business Administration, New York University, New York (January).

PADBERG, M. W., AND M. R. RAO. 1980c. The Russian Method and Integer Programming, GBA Working Paper, New York University, New York (January).

PICKEL, P. F. 1979. Some Improvements to Khachiyan's Algorithm in Linear Programming, Math. Department, Polytechnic Institute of New York, Farmingdale, N.Y. (December).

POLYAK, B. T. 1967. A General Method for Solving Extremum Problems. *Doklady Akademiia Nauk SSSR* **174,** 33–36 (translated in *Soviet Mathematics Doklady* **8,** 593–597, 1967).

POLYAK, B. T. 1969. Minimization of Unsmooth Functionals. *Zhurnal Vychisditel'noi Matematiki i Matematicheskoi Fiziki* **9,** 509–521 (translated in *USSR Computational Mathematics and Mathematical Physics* **9,** 14–29, 1969).

POLYAK, B. T. 1978. Subgradient Methods: A Survey of Soviet Research. In *Nonsmooth Optimization, IIASA Proceedings*, Vol. 3, C. Lemarechal and R. Mifflin, (eds.). Pergamon Press, Oxford.

SHOR, N. Z. 1964. On the Structure of Algorithms for the Numerical Solution of Optimal Planning and Design Problems, dissertation, Cybernetics Institute, Academy of Sciences of the Ukrainian SSR, Kiev.

SHOR, N. Z. 1968. The Rate of Convergence of the Generalized Gradient Descent Method. *Kibernetika* **4**(3), 98–99 (translated in *Cybernetics* **4**(3), 79–80, 1968).

SHOR, N. Z. 1970a. Utilization of the Operation of Space Dilatation in the Minimization of Convex Functions. *Kibernetika* **6**(1), 6–12 (translated in *Cybernetics* **6**(1), 7–15, 1970).

SHOR, N. Z. 1970b. Convergence Rate of the Gradient Descent Method with Dilatation of the Space. *Kibernetika* **6**(2), 80–85 (translated in *Cybernetics* **6**(2), 102–108, 1970).

SHOR, N. Z. 1976. Generalized Gradient Methods of Nondifferentiable Function Minimization and Their Application to Problems of Mathematical Programming. *Ekonomika i Matematicheskie Metody* **12,** 337–356.

SHOR, N. Z. 1977a. Cut-off Method with Space Extension in Convex Programming Problems. *Kibernetika* **13**(1), 94–95 (translated in *Cybernetics* **13**(1), 94–96).

SHOR, N. Z. 1977b. New Development Trends in Nondifferentiable Optimization. *Kibernetika* **13**(6), 87–91 (translated in *Cybernetics* **13**(6), 881–886, 1977).

SHOR, N. Z., AND V. I. GERSHOVICH. 1979. Family of Algorithms for Solving Convex Programming Problems. *Kibernetika* **15**(4), 62–67 (translated in *Cybernetics* **15**(4), 502–507, 1979).

SHOR, N. Z., AND N. G. ZHURBENKO. 1971. A Minimization Method Utilizing the Operation of Space Expansion in the Direction of the Difference of Two Successive Gradients. *Kibernetika* **7**(3), 51–59 (translated in *Cybernetics* **7**(3), 1971).

SKOKOV, V. A. 1974. Note on Minimization Methods Employing Space Stretching. *Kibernetika* **4**(4), 115–117 (translated as *Cybernetics* **4**(4), 689–692, 1974).

TELGEN, J. 1980. On Relaxation Methods for Systems of Linear Inequalities,

Technical Report, Management Science Program, University of Tenn. (January).

TODD, M. J. 1979. Some Remarks on the Relaxation Method for Linear Inequalities, Technical Report 419, School of Operations Research and Industrial Engineering, Cornell University, Ithaca, N.Y.

TODD, M. J. 1980. Minimum Volume Ellipsoid Containing Part of a Given Ellipsoid, Technical Report No. 468, School of Operations Research and Industrial Engineering, Cornell University, Ithaca, N.Y.

WOLFE, P. 1975. A Method of Conjugate Subgradients for Minimizing Nondifferentiable Functions. *Math. Program. Stud.* **3,** 145–173.

WOLFE, P. 1980. A Bibliography for the Ellipsoid Algorithm, IBM Research Center Report (July 7).

ZADEH, N. 1973. A Bad Network Problem for the Simplex Method and Other Minimum Cost Flow Algorithms. *Math. Program.* **5,** 255–266.