MATICS OF OPERATIONS RESEARCH
No. 2, May 1980
U.S.A.

A SURVEY AND COMPARISON OF METHODS FOR FINDING ALL VERTICES OF CONVEX POLYHEDRAL SETS*

T. H. MATHEISS† AND DAVID S. RUBIN‡

This paper surveys the literature on methods for finding all vertices of convex polytopes, ontrasting the main features of each method and providing computational results for epresentative methods.

oduction. Convex polytopes and other three-dimensional solids were studied ancients. However it was not until Euler's classic theorem (1752) relating the er of vertices, edges and faces of three-dimensional polytopes that a significant dealing with the combinatorial properties of convex polyhedral sets was ered. Since that time theoretical interest has waxed and waned several times. programming and other problems in the decision sciences have rekindled t in the combinatorial properties of polyhedra. Dantzig's simplex method d concern on the extremal properties of polyhedra. Results relating the number ices to the number of faces, the establishment of a least upper bound on the r of vertices, and a formula for the expected number of vertices for a given r of faces were and are being sought because of their practical importance for tational purposes.

hedra pervade the modeling process. They exist wherever a linear program is a ble model of reality, and in many other contexts such as mathematical mming, game theory, statistical decision theory, mathematical biology, and ography as well.

Applied Mathematics, Operations Research, Computer Science and Manage-ience literature contains several algorithms for obtaining all vertices of convex dral sets, and in particular, for convex polytopes (which are bounded convex dra). This study surveys that literature, presenting the intuition which motivates ous approaches, discussing some computational aspects of each and present- results of computational experience with the most promising vertex finding ures. There are also algorithms for finding all the facets of convex polytopes 8], [56]. However, enumerating the facets of a polytope is equivalent to ating the vertices of its polar (see Grünbaum [28, pp. 46–48].)

roblem we are addressing is to find all vertices of a given convex polyhedron, d as the intersection of a finite number of hyperplanes and closed half-spaces. thod of solving the problem must provide for (1) finding each vertex and (2) ing when all vertices have been found. It is desirable to accomplish this in an way.

and Witzgall [57] give a pivoting algorithm for finding all the vertices of a olytope. However, their algorithm assumes that the polytope is given as the

October 23, 1978; revised June 5, 1979.
subject classification. Primary 90C05. Secondary 52A25.
subject classification. Main: Sets. Cross reference: Linear, programming.
dex 1978 subject classification. Primary: 637 Programming, Linear algorithms. Secondary: 436

Convex polyhedra, finding all vertices, survey.

of Alabama, Tuscaloosa.
of North Carolina, Chapel Hill.

0364-765X/80/0502/0167$01.25

convex hull of a finite set of points, and it determines which of those points are vertices (and hence necessary for the description of the convex hull) and which are not vertices (and hence redundant). This problem is equivalent to eliminating the redundant constraints in our problem, and this latter problem has also been addressed by several authors [16], [27], [32], [35], [46], [49], [52]. As Mattheiss has shown [37], algorithms for finding all vertices can also be used to eliminate redundant constraints, but we will not discuss that problem in this paper.

Since the simplex algorithm moves from one vertex to an adjacent vertex, an apparently attractive approach to the problem is to use that algorithm in an iterative fashion to find a path which contains all vertices and passes through each one of them only once. Such a path is called a Hamiltonian path. If a 1-polytope is considered, the solution is simply the two endpoints. If a 2-polytope is considered, the solution is easily obtained by beginning at some vertex and then traversing the perimeter of the polytyope by successive exchanges of 1-faces in the linear system defining the polytope. This occurs because the 1-faces of 2-polytopes are naturally ordered into one cycle. These views of the problem are deceptively simple. It was shown by Brown [4] (also see the example of Tutte [54]), that there are 3-polytopes for which Hamiltonian paths do not exist and therefore such paths do not exist for n-polyhedra in general. Therefore any method of solving the problem by this approach must construct a path that either visits a subset of the vertices more than once or visits points in R^n (or higher dimensions) which are not on the boundary of the polyhedron. However, it must be pointed out that Barnette [2] has conjectured that all simple (i.e., nondegenerate) 4-polytopes do have Hamiltonian paths, so there is yet the possibility that an algorithm for vertex enumeration via Hamiltonian paths may be found. Barnette's conjecture has been proved for some prisms [45].

More ponderous than the practical questions raised by the nonexistence of a Hamiltonian path is the sheer volume of computation involved in obtaining the vertices. Several of the methods examined involve the generation and analysis of at least one simplex tableau for each vertex of the polyhedron. Let $\overline{V}(P)$ be a least upper bound on the number of vertices of a polytope P. The Upper Bound Conjecture gives $\overline{V}(P)$ in terms of m (the number of $(n-1)$-faces) and n (the dimension of the space) as:

$$\overline{V}(P)=\binom{m-[(n+1)/2]}{m-n}+\binom{m-[(n+2)/2]}{m-n}$$

where [*] denotes the greatest integer function and $\binom{*}{*}$ denotes the familiar binomial coefficient. The bound $\overline{V}(P)$ is achieved by the cyclic polytopes studied by Gale [23] and others. Klee discusses the early history of the conjecture and of cyclic polytopes in [30], where he also proves the conjecture for all polytopes with $m \geqslant n^2/4 - 1$. The bound was shown to be sharp for all m and n by McMullen [39]. Even for relatively small problems, $\overline{V}(P)$ can be enormous, as shown in Appendix E.

Let $\underline{V}(P)$ be a greatest lower bound on the number of vertices. Grünbaum [28, p. 188] states the conjecture that for simple polytopes

$$\underline{V}(P)=(n-1)m-(n-2)(n+1)$$

which is proved by Barnette [3], so there are polytopes with relatively few vertices. Liebling [34a] asserts that there are many linear programming problems having large m and n and relatively small numbers of vertices. He calls such polytopes benevolent and credits the computational success of the simplex algorithm to their high frequency of occurrence in nature.

Practical considerations focus on the expected value of the number of vertices $E(V)$. Schmidt and Mattheiss [37a], [47] give several results for $E(V)$ based on 9,867 randomly generated 4-, 7-, and 10-polytopes. Related work has also been done by Grünbaum et al. [15a].

Algorithms for enumerating all vertices of a polyhedron can be divided into two classes: pivoting methods and nonpivoting methods. Some of the methods to be discussed assume that the polyhedron in question is a polytope. So long as the polyhedron is bounded below, i.e., there exists an r in R^n such that $x \geqslant r$ for all $x \in P$ (which will be true in the common case where the variables are restricted to be nonnegative), all unbounded edges can be truncated by the usual "regularization" device [9, p. 182] of adding a bounding constraint on the sum of the variables. Although we will not pursue the details, most of the methods are easily modified to handle unboundedness without regularization.

In order for a convex polyhedron to have any vertices, its lineality space (the largest linear subspace it contains) must have dimension 0. All the algorithms discussed assume this to be the case. This condition always holds when the defining constraints of the polyhedron include nonnegativity of the variables.

Most of the pivoting algorithms assume that the polyhedron is nondegenerate. However, all of them can be modified to handle degeneracy by using standard perturbation or lexicographic schemes [29, Chapter 6]. The nonpivoting schemes are unaffected by the presence of degeneracy.

§1 of this survey discusses pivoting methods, while §2 is devoted to nonpivoting methods. In §3, we present the results of a computational study of the methods of Balinski [1], Chernikova [12], Manas-Nedoma [36], and Mattheiss [37]. We assume that the reader is familiar with the simplex algorithm and the theory of convex polyhedral cones. A good survey of the latter is given by Gerstenhaber [25].

1. Pivoting methods. To facilitate the exposition, the Tucker [53] tableau and corresponding geometry will be employed when convenient, although it is clear that implementation might be facilitated by, for example, the revised simplex procedure.

In 1953, Charnes [8] presented the Spiral Method for Effecting a Grand Traversal. This technique applies the simplex of Dantzig to the Tarry procedure given in König [illegible] for resolving the labyrinth problem of the theory of graphs. This procedure appears to be computationally infeasible for computer processing due to enormous storage requirements imposed by the necessity of knowing, for every vertex, how often and in what direction the edges emanating from that vertex have been traversed. Also, the procedure requires that each edge is traversed twice and each vertex visited at least n times.

Balinski [1] alludes to a number of cutting plane methods suggested by the work of Gomory [26]. He concludes that these methods are not computationally feasibly in that the addition of more half-space requirements to eliminate the vertices already found creates additional "vertices" which are not vertices of the original polytope. Further, the additional constraints enlarge the set of inequalities which define the polytope and must be manipulated. Balinski also refers to a pseudolinear objective function technique which orders the vertices of the polytope in some way. He concludes that termination criteria and the fact that the pseudo-objective function does not order the vertices of the polytope in a path which could be followed by successive steps of the simplex method renders this approach computationally infeasible.

The vertices of 2-polytopes are particularly easy to find by the simplex method because they are naturally ordered into one cycle. In 1961, Balinski [1] published the first algorithm which was coded for a computer exploiting this idea. This algorithm finds all vertices of the polytope first by choosing a defining hyperplane say H_i. All vertices of the polytope which lie on H_i are found by fixing a face of a face of

(i) The distance between two vertices v_i and v_j is defined by $d(v_i, v_j) = c(x_i -$ (The z-row is not deleted in rule (i).)

(ii) Rule (iii) is modified so that the next index set to be selected from the list is which minimizes $d(v^*, v_j)$; where v^* represents the current vertex and j ranges ove unflagged index sets on the list.

The fact that each vertex has a value of the objective function associated wi allows list searching to be handled more efficiently than in the Manas and Ned algorithm. Contrarily, the ranking according to a decreasing sequence of obje function values may require an excessive amount of pivoting from one "side" o polytope to another, relative to the Manas and Nedoma procedure. Computati results on this innovation are not as yet available in published form.

The method of Burdet [5] determines the vertices of a polytope P as the 0-faces i "facial arborescence" of the polytope. The root of the tree is P itself, with each no at level $k(n \geqslant k \geqslant 0)$ corresponding to a k-face of P. At each node, a large number linear programs must be solved (one for each nonredundant constraint defining t boundary of its ancestor node) to determine the boundary of the current face o and the corresponding branches to the next lower level. The index sets correspondi to the branches are generated in lexicographically increasing order so that repetition may occur in the construction of the tree above level 1. All vertices of P a generated at least once and possibly as many as n times. Storage requirements a modest, consisting of two tableaux and the index sets defining the arborescence. T method was not considered further due to the enormous computational requireme of the linear programs and the pivots necessary for multiple visits to vertices.

Recently, Dyer and Proll [17] have given a pivoting algorithm for determining a vertices of a convex polyhedron. The algorithm constructs a spanning tree of th edge-vertex graph of the polyhedron, starting with an arbitrary node as the root. each arc of the graph has length 1, we say that two nodes are *k-neighbors* if th shortest path joining them in the graph has length k. Two nodes are *adjacent* if the are 1-neighbors. A node has *height k* if it is a k-neighbor of the root node.

The algorithm may be described briefly as follows:

1. $k \leftarrow 0$.
2. $k \leftarrow k + 1$.
3. Find all nodes with height k. If there are none, all feasible bases have been found, so stop. Otherwise go to step 2.

The algorithm is finite because the graph is finite and connected, and it uses the fact that every node adjacent to a node with height k has height $k - 1, k$, or $k + 1$; and conversely, every node with height $k + 1$ is adjacent to at least one node with height k.

Given this description, it is clear that this algorithm does not present any new basic approach to the problem, for it is a standard way to find spanning trees and shortest paths. For its use in vertex enumeration, see Remez and Shteinberg [44]. What is new, however, is the way in which the algorithm is implemented, in particular its use of the revised simplex method and its data organization for performing a breadth-first search of the spanning tree. Dyer and Proll report only limited computational experience with their algorithm, but, as we have reported elsewhere [38], it appears to be computationally inefficient.

All of the methods referred to above attempt to deal with the polytope in the same dimensional space in which it is described. In 1973 Mattheiss [37] gave an entirely new approach to the problem. Geometrically stated, his method embeds the given polytope in a one-higher dimensional Euclidean space. The projections into the original space of the additional vertices and edges formed by the embedding process lie in the interior of the polytope and form a connected graph. The embedding process also

ciates a number with each interior node which enables the construction of a ning tree for all of the interior points. The tree so constructed has the vertices of polytope as *termini ad quem*. Each interior node is represented by a simplex leau, all of which must be produced and analyzed. The actual simplex tableaux esponding to the vertices of the polytope need not be produced *per se* since they be obtained from the tableau representing the appropriate interior node. Appen- C presents the Mattheiss Algorithm.

he efficacy of the method rests on the condition that the number of interior nodes the spanning tree is less than the number of vertices of the polytope. Of 5,237 ndomly generated (see Schmidt and Mattheiss [47]) 4-polytopes, 12 were found hich violated the condition. In addition, 3,373 7-polytopes and 453 10-polytopes re generated; none of which violated Mattheiss' condition. Klee [31] has given an tended discussion of this condition.

2. Nonpivoting methods. All of the nonpivoting methods can be viewed as riants of the Double Description Method of Motzkin, Thompson, Raiffa, and hrall [41]. As Duffin [16] and Dantzig and Eaves [14] have pointed out, these ethods are dual to the Fourier-Motzkin elimination technique for the solution of ear inequality systems [21], [40].

It should be pointed out that many of these methods are originally stated in terms f finding all the extreme rays of convex polyhedral cones. However, $\bar{x}$ is a vertex of he polyhedron $P = \{x \mid Ax \leqslant b\}$ if and only if $((\bar{x}, 1))$ is an extreme ray of the cone $= \{(x, \xi) \mid -Ax + b\xi \geqslant 0, \xi \geqslant 0\}$. Here we have used $((\bar{x}, 1))$ to denote $\{(\lambda\bar{x}, \lambda) \mid \lambda \geqslant 0\}$.

These methods are geometrically motivated, but their algebraic foundations are discussed by Bürger [6] (for cones) and more recently by Galperin [24] (for polyhedra). Because the geometric foundation is so intuitively appealing, our presentation here will be geometric. Suppose we have a polytope P, whose vertices are already known. Suppose that P' is obtained from P by adding another constraint (that is P' is the intersection of P and a hyperplane H or a closed half space H^+.) Then the vertices of P' are some of the vertices of P (those on H or in H^+) and certain convex combinations of vertices of P in H^+ with other vertices of P in H^-. The weights in these convex combinations are chosen so that the new vertices all lie on H.

Uzawa [55] has given an algorithm based on this observation. His algorithm finds all the vertices of a polyhedron, but it also produces points which are not vertices. Let $x_1, \ldots, x_k$ be all the vertices of P, and now consider the additional constraint $x \in H^+$. Suppose that $\{x_1, \ldots, x_p\} \subseteq H^+ \setminus H, \{x_{p+1}, \ldots, x_q\} \subseteq H$, and $\{x_{q+1}, \ldots, x_k\} \subseteq H^- \setminus H$. Then for the vertices of P' Uzawa lists $x_1, \ldots, x_q$ plus a point of the form $\hat{x} = \lambda x_i + (1 - \lambda)z_j$ for each $i \in \{1, \ldots, p\}$ and $j \in \{q + 1, \ldots, k\}$. However, such an $\hat{x}$ is a vertex of P' if and only if x_i and x_j are adjacent vertices of P (i.e., they determine an edge of P.) Because of this, Uzawa's method will be faster than methods which check for adjacency, but will require more storage. Furthermore, some way must be found to purge the final list of the nonvertex points it contains. (In a similar fashion, Fourier-Motzkin elimination leads to redundant constraints in the course of eliminating variables. Kohler [32] and Duffin [16] have discussed techniques to avoid the creation of such redundant constraints.)

Rather than eliminate nonextreme points at the end of the process, the Double Description Method and its variants [6], [10–12], [24], [27], [32], [34] avoid generating them in the first place. To do this, it is necessary to determine when two vertices x_i and x_j are adjacent. Now an edge is a 1-dimensional face of a polyhedron, and in general a d-dimensional face of a polyhedron in n space is the intersection of $n - d$ linearly independent hyperplanes from the constraints defining the polyhedron. Thus

x_i and x_j are adjacent if and only if they both satisfy at least $n-1$ of the defin constraints (inequalities and/or equalities) as equalities, and exactly $n-1$ of th constraints are linearly independent. This condition is tedious to verify computatio ally, but fortunately a simpler condition can be used to characterize edges: x_i and are adjacent if and only if no other vertex lies on the face of P which they determ As we shall see below, this condition is easy to verify. (For polytopes defined $P = \{x \mid Ax = b, x \geq 0\}$, Murty [42a] has characterized the adjacency of two vert in terms of the rank of the set of columns of A corresponding to variables which positive at the two vertices. Although this condition is simple to verify in the cont of pivoting algorithms, in the current context it would require considerable effort reduce the corresponding submatrix of A to echelon form to determine its rank.)

The Double Description Method considers the cone $C = \{x \mid Ax \leq 0\}$ and de scribes C as the direct sum $\hat{C} + L$, where $\hat{C}$ is a pointed cone and L is the lineali space of C (i.e., the largest subspace contained in C.) For $\hat{C}$ the method finds all th extreme rays, and for L it finds a basis. In the case where the constraints $Ax \leq$ subsume the constraints $x \geq 0$, then L has dimension 0 and the Double Descriptio Method is identical to the procedure given by Chernikova [12]. That algorithm given in Appendix D.

Step 3 of the algorithm is the determination of whether two edges of C (or vertice of P) are adjacent. $I_1(s,t)$ identifies the constraints which define the minimal dimen sion face of C that contains the two edges in question $((l_s)$ and $(l_t))$. If $I_1(s,t) = \emptyset$ then C itself is that face, so the edges are not adjacent (unless C has only those two edges). Step 3b sees if any other edge of C is on this minimal face. If so, the face has dimension > 2, and (l_s) and (l_t) are not adjacent; if not, the face has dimension 2 and (l_s) and (l_t) are adjacent. When the algorithm is programmed, Step 3 can be efficiently implemented through the use of binary coded data (indicating which constraints are tight on each edge) and fullword logical operations. The program used in our computational experiments employs these devices. It also uses a device not contained in Chernikova's description of the algorithm. In n-space it takes $n-2$ independent equations to determine a face of dimension 2. Thus if $I_1(s,t)$ does not contain at least $n-1$ elements (because C is a cone in $(n+1)$-space), then (l_s) and (l_t) are not adjacent. Our program implements this test before the test of step 3b, again making efficient use of the binary coded data. This test has been used previously by Kohler [32]. It also appears in [27], where Greenberg incorrectly asserts that it is a necessary and sufficient test for adjacency. This error was pointed out by Sherman [50].

We now show how to handle equality constraints, which the Double Description Method does not explicitly consider. Equality constraints of the form $\sum a_{ij}x_j = b_i$ can be incorporated into the algorithm by splitting them into two inequality constraints $\sum a_{ij}x_j \leq b_i$ and $-\sum a_{ij}x_j \leq -b_i$. However, if these two rows were processed sequentially, it is easy to see that the effect of this is identical to including only one row for the constraint, and modifying step 2 by redefining R to be $\{j \mid y_{rj} = 0\}$. This is precisely the result given by Chernikova in an earlier paper [11].

The situation when the constraints $x \geq 0$ are not present is somewhat more complex, and we refer the interested reader to the Double Description Method [41, pp. 67–69] and the work of Kuznetsov [34] and Chernikov [10]. Since we are interested in vertex enumeration, we assume that our polyhedron has at least one vertex, and hence that its lineality space has dimension 0. Then by translation and the standard "regularization" technique we can bound our polyhedron and translate it to the nonnegative orthant. The vertices of the original polyhedron will correspond to those vertices of the resulting polytope not on the "regularizing" hyperplane, while its extreme rays will correspond to those vertices of the polytope on the "regularizing"

erplane. For this reason we do not discuss here the case of variables not restricted gn.

3. **Computational results.** From a computational point of view, the most impor criteria for comparing algorithmic performance are accuracy, time, and storage irements. According to these criteria, the methods of Balinski, Manas-Nedoma Mattheiss appeared to be representative of the pivoting methods and Chernikova's hod representative of the nonpivoting methods. These algorithms were pro mmed in FORTRAN and applied to the same set of sample problems. The orithms of Balinski, Manas-Nedoma and Mattheiss were programmed in double- ecision. All computations were carried out on an IBM 370/168 operating under SMVT. One virtual machine comprising 512K was the storage limitation for all ethods.

The sample was obtained from a random polytope generator (see Schmidt and Mattheiss [47]). The sample design is given in Table 1 along with the actual yield of olytopes from the generator. If an n-polytope having m facets was requested of the enerator, an n-polytope having $k \leq m$ facets was supplied by the generator, because ome of the m constraints were redundant. Redundant constraints were not elimi ated before the algorithms were run. In addition, two transportation type polytopes

TABLE 1

Sample Design and Yield of Random Polytopes from the Generator

m n	2		5		8		11		14		17		20	
4	15	(10)	—		—		—		—		—		—	
5	7		—		—		—		—		—		—	
6	10		—		—		—		—		—		—	
7	3		12	(10)	—		—		—		—		—	
8	2		5		—		—		—		—		—	
9	3		1		—		—		—		—		—	
10	—		3		11	(10)	—		—		—		—	
11	—	(10)	2		2		—		—		—		—	
12	—		4		4		—		—		—		—	
13	—		1		2		10	(10)	—		—		—	
14	—		6	(10)	2		1		—		—		—	
15	—		2		1		—		—		—		—	
16	—		—		3		2		10	(10)	—		—	
17	—		—		3	(10)	1		1		—		—	
18	—	(10)	2		2		1		2		—		—	
19	—		1		—		3		—		10	(10)	—	
20	—		1		1*B*		6	(10)	2		—		—	
21	—		—	(10)	1		—		1		1		—	
22	—		—		3		—		—		—		10	(10)
23	—		—		—		—		7*N*	(10)	1		—	
24	—		—		1	(10)	1		—		1		1	
25	—	(10)	—		2*C*		2		1		—		—	
26	—		—		1		2		—		7	(10)	2	
27	—		—		1		1	(10)	—		—		—	
28	—		—	(10)	—		—		1		—		—	
29	—		—		—		—		2		—		1*M*	(4)
30	—		—		—	(10)	—		1	(8)	—		—	

NOTE. Numbers in parentheses indicate that m constraints were requested from the generator. Numbers not in parentheses are the number of polytopes obtained having k relevant constraints.

B indicates the largest problem completed by Balinski's algorithm.

C indicates the largest problem completed by Chernikova's algorithm.

M indicates the largest problem completed by Mattheiss' algorithm.

N indicates the largest problem completed by Manas-Nedoma's algorithm.

of sizes 24 × 14 and 20 × 11 were included in the sample. All problems except larger transportation problem were nondegenerate.

Table 1 also shows the largest size problem from the sample which was successfu completed by the respective methods. In each case, several unsuccessful attempts we made to complete problems of larger size. The largest problem handled by Balinsk method was 20 × 8. Larger problems consumed time somewhere in excess of 6 seconds. The largest problem successfully completed by Chernikova's method w 25 × 8. The storage requirement for larger problems exceeded the 512K allotted. Th largest problem completed by the Manas-Nedoma algorithm was 23 × 14. For lar size problems, storage requirements exeeded 512K. Mattheiss' algorithm completed of the problems in the design, the largest of which was 29 × 20.

Scattergraphs for the methods of Balinski, Chernikova, Manas and Nedoma, an Mattheiss are displayed in Figure 1 through Figure 4, respectively. Least squa

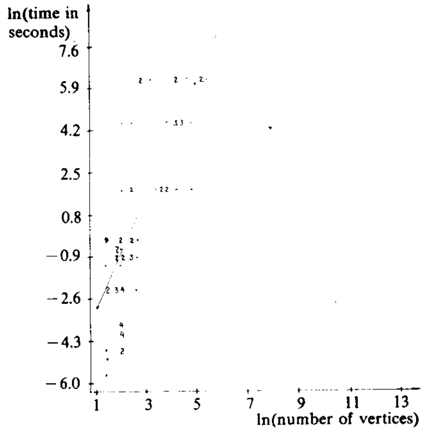

FIGURE 1. Scattergram for Balinski's Method. ln(time in seconds) = −5.345 + 2.272 ln(number of vertices).

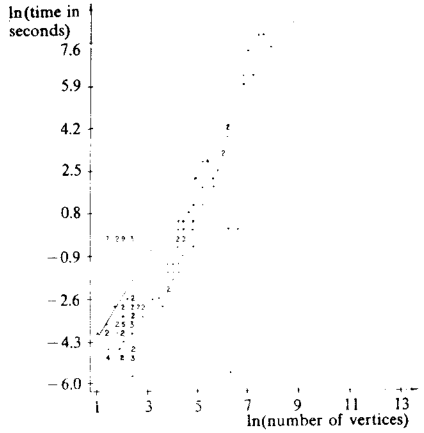

FIGURE 2. Scattergram for Chernikova's Method. ln(time in seconds) = −5.589 + 1.418 ln(number of vertices).

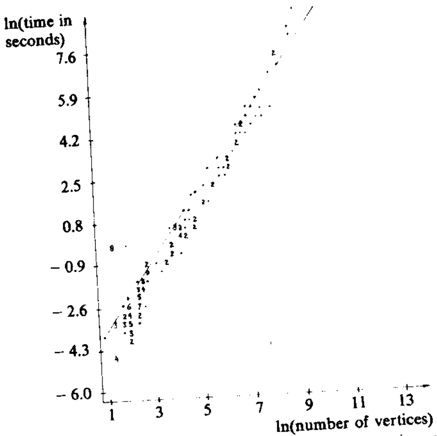

FIGURE 3. Scattergram for Manas-Nedoma's Method of ln(time in seconds) = −5.046 + 1.379 number of vertices).

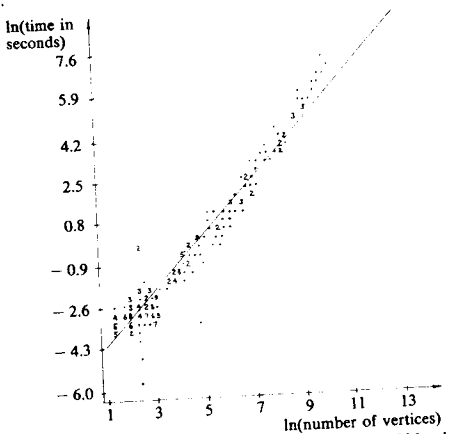

FIGURE 4. Scattergram for Mattheiss Method ln(time in seconds) = −5.386 + 1.146 ln(number of vertices).

log log regression results are given in Table 2 and shown in Figure 1 through Figure 4. (On some of the test problems, Balinski's algorithm gave incorrect results. These problems were excluded from the regression. The difficulty arises from the way the algorithm handles tableaux which are "not acceptable" (see [1, pp.78–79]). Dyer and Proll [18] have constructed a simple example showing the error in the algorithm and have shown how to correct it. This result was not available to us when our computational experiment was run; however, the sketch in Appendix A uses their correction at step iv.) Table 2 provides a comparison of the log log regression results which are graphed in Figure 5 for convenience. These results indicate that for very small problems the Chernikova algorithm slightly outperforms the others tested against the time criterion. However, this initial advantage rapidly fades. Since the intercept term b_0 is nearly the same for all four methods, the superiority of the algorithms can be determined by the b_1 term. Ranked according to the time criterion, the four algo-

TABLE 2

Regression Results on a Set of Randomly Generated Polytopes.
ln(*time in seconds*) = $b_0 + b_1$ln(*number of vertices*).

Algorithm	b_0	b_1	R^2	Error of Estimate	Sample Size
Balinski	− 5.345	2.272	0.605	2.160	80
Chernikova	− 5.589	1.418	0.633	1.853	118
Manas & Nedoma	− 5.046	1.379	0.891	1.005	164
Mattheiss	− 5.386	1.146	0.947	0.672	214

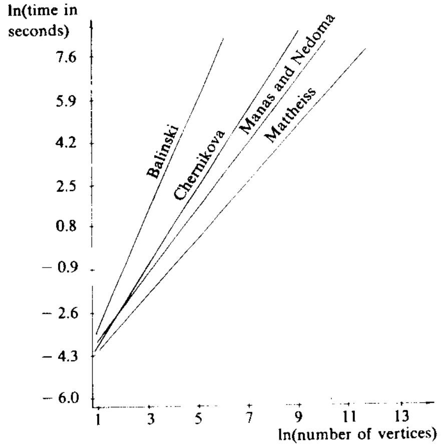

FIGURE 5. Comparison of times for all four methods.

rithms are, from most desirable to least desirable; Mattheiss, Manas and Nedoma, Chernikova and Balinski.

In a related study, Dyer and Proll [19] have also looked at the algorithms of Balinski, Chernikova, Greenberg [27], and Manas-Nedoma, as well as their own procedure. Their paper discusses in detail some of the decisions they made in implementing these algorithms and outlines the storage requirements for each of them. Their computational study was more limited than ours, looking at only 20 test problems the largest of which were 16 × 10 and 17 × 5. None of their test problems had more than 250 vertices. From their limited work, they rank the algorithms in rough order of efficiency (by the time criterion): Dyer-Proll, a modified Manas-Nedoma, Greenberg, and finally Balinski and Chernikova. (The last two could be not be uniformly ranked with respect to each other.) Except for the relative positions of Balinski and Chernikova, this ranking is consistent with our results. (Dyer and Proll [20] report they have modified Greenberg's algorithm to correct the error mentioned in §2 above, but it is not clear whether the change was made prior to the results reported in [19].

Dyer and Proll emphasize [19, p. 24], that they carefully refined the program for their own method in the course of the study, but did not similarly refine the other programs. In a similar fashion, we have recoded the Chernikova algorithm since completing the computational work reported here, and it nows appears to run about twice as fast as it did before. It may be possible to improve our other codes as well. Because of this difference in attention to coding details, and because of differences in machine speeds, both their results and ours should not be seen as conclusive, but rather as benchmarks for further research.

Appendix A. A sketch of Balinski's algorithm. Let $P = \{x \mid Ax \leqslant b\}$ be a convex polytope in R^n, where A is a real $m \times n$ matrix, $m > n$, x and b are conformable real vectors. Introduce nonnegative slack variables so that the system can be written

$$A(-x) + b = y \geqslant 0. \tag{A-1}$$

DEFINITIONS. The *index* of a variable y_i is the number of times which y_i was nonbasic at a vertex of X.

B_j, $j = 0, \ldots, n$ are sets for indexing the variables. B_0 contains all of the x-variables. B_j contains the slack variables corresponding to $(j-1)$-faces of P all of whose vertices have been found, $j = 2, \ldots, n$. B_1 contains the slack variables not contained in $B_0, B_2, \ldots, B_n$.

The convex polyhedral set P_k is formed by deleting the $y_i \geqslant 0$, $i = 1, \ldots, k$, from (A-1) whose vertices have already been found.

A tableau is *acceptable* if all elements in the constant column and in rows labeled $y_i \in B_1 \cup B_2$ are nonnegative. An acceptable tableau corresponds to a point which is a vertex of the set $H_r \cap P_k$, where H_r is the hyperplane being traversed. If all $y_i \geqslant 0$ then the tableau corresponds to a vertex of P.

Beginning with (A-1) and making all x-variables basic leads to the tableau representing a general step of the algorithm.

$$
\begin{array}{r|c|cc|cccc|}
 & 1 & -y_1 & -y_2 & \overbrace{-y_3 \quad \cdots \quad -y_{n-1} \quad -y_n}^{\text{allowing no pivots fixes a 2-face of } P} & & & \\
\hline
x_i \in B_0 & & & & \text{No Pivots} & & & \\
\hline
y_i \in B_1 & & & & & & & \\
y_i \in B_2 & & & & \text{No Pivots In } k\text{-th Stage} & & & \\
y_i \in B_3 & & & & & & & \\
\hline
\vdots & & & & \text{No Pivots} & & & \\
y_i \in B_{n-1} & & & & & & & \\
y_i \in B_n & & & & & & & \\
\hline
\end{array}
\tag{A-2}
$$

Assume that all vertices belonging to the $(n-1)$-faces of P whose corresponding $y_i \in B_n$ have been found. Further, assume that all vertices belonging to the $(n-2)$-faces of P whose corresponding $y_i \in B_{n-1}$ have been found, etc., until some particular 2-face of P_k has been fixed. At each level of this process the nonbasic variable with the highest index is selected to be fixed.

A particular 2-face of P_k is traversed by a sequence of acceptable tableaux (points) obtained from selected positive pivot elements in only two columns of (A-2), say those labeled $-y_1$ and $-y_2$ in rows labeled $y_i \in B_i$.

If a stage is begun with an acceptable tableau then either:

(i) a circuit around the 2-face of P_k can be completed;

(ii) several pivots can be completed "in one direction," the initial point reassumed, and possibly several pivots can be completed in the "other direction";

(iii) no pivots are possible in either direction and there is but one acceptable point in the stage.

Bookkeeping requirements include moving the nonbasic pivot variable of the second acceptable tableau of the stage from B_1 into B_2 in the third tableau and similarly in subsequent tableaux. Appropriate labeling is required to recognize the initial point of the stage.

If a stage is begun with an unacceptable tableau then [18]:

(iv) At each pivotal step, we look for rows (of the two-dimensional tableau) which are manifestly infeasible, while pivoting to keep nonnegative all currently nonnegative basic variables. The pivot column is chosen so that it contains at least one negative element in a row with a negative right hand side, and it is permitted to pivot on a negative element in such a row (provided this does not change the sign of any positive basic).

(v) If no element exists as in (iv), this stage is complete.

The end of the kth stage occurs when all acceptable points of H_r have been found. Then a pivot is chosen whose column is labeled $-y_3$ and whose row is not labeled B_1 or $B_3, \ldots, B_n$. The row of the pivot is chosen such that the least number of elements of the column of constants corresponding to variables of B_1 and B_2 are negative. If such a pivot is possible, perform the pivot, move y_3 to B_3 and make B_2 null by moving all of its variables into B_1. If all elements of the column labeled $-y_3$ which are in rows of B_1 and B_2 are zero then no such pivot exists and all of the vertices of the fixed 3-face of P have been found.

In general, a pivot is performed whose column label is $-y_r$ and whose row is not labeled B_0 or $B_r, \ldots, B_n$. The variable y_r is moved into B_r and the sets $B_2, \ldots, B_{r-1}$ are made null by moving their elements into B_1.

The process terminates if either all elements in the column labeled $-y_r$ and not in rows labeled B_0 or B_r are zero or if $r = m - n$.

Appendix B. Manas and Nedoma's algorithm. Let $P = \{x \mid Ax \leqslant b\}$ be a convex polytope in R^n. To initialize the algorithm solve the following linear program.

$$\begin{cases} \begin{pmatrix} A & I \\ c & 0 \end{pmatrix}\begin{pmatrix} x \\ s \end{pmatrix} = \begin{pmatrix} b \\ z \end{pmatrix}, \\ \text{minimize } z \end{cases} \tag{B-1}$$

where A is a real $m \times n$ matrix, $m > n$; I is an m-dimensional identity matrix; with b, s, x and z conformable real vectors, and c a conformable vector of ones.

The form of the optimal tableau for (B-1) is as follows:

	1	− (nonbasic slacks)
z	z^*	$\geqslant 0$
x	x^*	no pivots allowed
(basic slacks)	$s^* \geqslant 0$	

(B-2)

The vertex enumeration algorithm proceeds as follows.

(i) The initial vertex is x^* and the z-row is deleted from (B-2).

(ii) Form a list of nonbasic slack variable index sets. The initial set is that found in (B-2) together with those constructed by finding the pivot element in each column of (B-2) which exchanges slack variable indexes.

The general step of the algorithm is as follows.

(iii) Select an unflagged index set from the list having minimum distance from the current nonbasic index set. Two different index sets have the distance $d \leqslant n$ if exactly d components of one of them are different from the other. Flag the new index set. Produce the corresponding tableau and output the vertex x.

(iv) For each column in the new tableau, find the pivot element and place the corresponding slack variable index set on the list if it is not already on the list.

(v) Perform (iii) and (iv) until all index sets are flagged.

Appendix C. Mattheiss' algorithm. Let $P = \{Ax \leqslant b\}$ be a convex polytope in R^n. To initialize the algorithm embed P in R^{n+1} and solve the following linear program.

$$\begin{cases} Ax + ty + Is = b, \\ \text{maximize } y \end{cases} \tag{C-1}$$

where A is a real $m \times n$ matrix; I an m-dimensional identity matrix; with b, s and x conformable real vectors, and y a real variable; the real vector t has elements $= (\sum_{j=1}^{n} a_{ij}^2)^{1/2}$.

The form of the optimal tableau for this LP is as follows.

	1	− (nonbasic slacks)
y_0	$y^* > 0$	$\geqslant 0$
x	x^*	no pivots allowed
y	$y^* > 0$	
(basic slacks)	$\geqslant 0$	

(C-2)

The vertex enumeration algorithm proceeds as follows.

(i) For each column of (C-2) having a nonnegative dual variable, find the pivot element in that column. A pivot element must occur in either the y-row or in some slack row.

(ii) If the pivot element in the column under consideration is in the y-row, a vertex of P is obtained by performing a partial pivot operation on (C-2) with this pivot. Only the x-portion of the 1-column need be transformed.

(iii) If the pivot element in the column under consideration results in an exchange of slack labels:

(a) Compute the new value of y and form the index set of nonbasic slacks which identifies the new tableau based on the pivot being considered.

(b) Construct an ordered list of the sets of nonbasic slack indices defining the tableaux yet to be examined. The list is maintained in accordance with the magnitude of y ranked from high to low. If a candidate slack index set is already on the list, it is simply discarded.

(c) Flag all slack variables which are members of any slack index set of the list. At the end of the procedure, the flagged slacks identify the set of boundary constraints of P. Those not flagged are irrelevant for P.

(iv) When all columns of the current tableau have been analyzed,

(a) Execute a pivot (if possible) on a slack label exchanging element of (C-2) having a nonnegative dual variable. Otherwise,

(b) Select the slack index set from the top of the list and obtain the corresponding tableau. These tableaux have the same form as (C-2) except that certain dual variable values may be negative. Delete the current slack index set from the list. Return to (i).

(v) Perform (i) through (iv) until the list is empty.

Appendix D. Chernikova's algorithm. Consider the polyhedron $P = \{x \mid Ax \leqslant b, x \geqslant 0\}$ (where A is $m \times n$) and the related cone $C = \{(x, \xi) \mid -Ax + b\xi \geqslant 0, x \geqslant 0, \xi \geqslant 0\}$. To find all the vertices and extreme rays (unbounded edges) of P, we find all the extreme rays of C. Those with $\xi > 0$ correspond to vertices of P, those with $\xi = 0$ correspond to extreme rays of P.

Consider the matrix $\left(\begin{smallmatrix} -A & b \\ I & \end{smallmatrix}\right)$, where I is an $(n+1) \times (n+1)$ identity matrix. We give a series of transformations of this matrix which generates the solution. At any stage of the process we denote the old matrix by $y = \left(\begin{smallmatrix} U \\ L \end{smallmatrix}\right)$, and the new matrix being generated denoted $\overline{Y}$. The matrices U and L will always have m and $n+1$ rows, respectively; however, they will in general not have $n+1$ columns. They will have more than $n+1$ columns in most cases, but if C lies in some subspace of R^{n+1} they may have fewer than $n+1$ columns. For $(x, \xi) \in R^{n+1}$, we use the symbol $((x, \xi))$ to denote the ray $\{(\lambda, x, \lambda\xi) \mid \lambda \geqslant 0\}$.

The algorithm is as follows:

(0.0) If any row of U has all components negative, then $(x, \xi) = 0$ is the only solution.

(0.1) If all the elements of U are nonnegative, then the columns of L are the edges of C, i.e., the ray $(l_j) = ((x, \xi)) = \{\lambda l_j \mid \lambda > 0\}$ is an edge of C; here l_j denotes the jth column of L.

(1) Choose the first row of U, say row r, with at least one negative element.

(2) Let $R = \{j \mid y_{rj} > 0\}$. Let $v = |R|$, i.e., the number of elements of R. Then the first v columns of the new matrix, $\bar{Y}$, are all the y_j for $j \in R$, where y_j denotes the jth column of Y.

(2′) If Y has only two columns and $y_{r1} y_{r2} < 0$, adjoin the column $|y_{r2}|y_1 + |y_{r1}|y_2$ to the $\bar{Y}$ matrix. Go to step 4.

(3) Let $s = \{(s, t) \mid y_{rs} y_{rt} < 0,\ s < t\}$, i.e., the set of all (unordered) pairs of columns of Y whose elements in row r have opposite signs. Let I_0 be the index set of all nonnegative rows of Y. For each $(s, t) \in S$, find all $i \in I_0$ such that $y_{is} = y_{it} = 0$. Call this set $I_1(s, t)$. We now use some of the elements of S to create additional columns for $\bar{Y}$:

(a) If $I_1(s, t) = \emptyset$ (the empty set), then y_s and y_t do not contribute another column to the new matrix.

(b) If $I_1(s, t) \neq \emptyset$, check to see if there is a u not equal to either s or t, such that $y_{iu} = 0$ for all $i \in I_1(s, t)$. If such a u exists, then y_s and y_t do not contribute another column to the new matrix. If no such u exists, then choose $\alpha_1, \alpha_2 > 0$ to satisfy $\alpha_1 y_{rs} + \alpha_2 y_{rt} = 0$. (One such choice is $\alpha_1 = |y_{rt}|, \alpha_2 = |y_{rs}|$.) Adjoin the column $\alpha_1 y_t + \alpha_2 y_s$ to the new matrix.

(4) When all pairs in S have been examined, and the additional columns (if any) have been added, we say that row r has been "processed." Now let Y denote the matrix $\bar{Y}$ produced in processing row r, and return to step (0.0).

Appendix E.

Selected values of the least upper bound on the number of vertices.

m \ n	2	3	5	15	25
3	3	0	0	0	0
4	4	4	0	0	0
5	5	6	0	0	0
6	6	8	6	0	0
7	7	10	12	0	0
8	8	12	20	0	0
9	9	14	30	0	0
10	10	16	42	0	0
11	11	18	56	0	0
13	13	22	90	0	0
15	15	26	132	0	0
16	16	28	156	16	0
20	20	36	272	1584	0
25	25	46	462	0.3890E + 05	0
26	26	48	506	0.6365E + 05	26
50	50	96	2162	0.5396E + 08	0.3705E + 10
60	60	116	3192	0.2676E + 09	0.1045E + 12
70	70	136	4422	0.9836E + 09	0.1415E + 13
80	80	156	5852	0.2946E + 10	0.1199E + 14
90	90	176	7482	0.7604E + 10	0.7350E + 14
100	100	196	9312	0.1752E + 11	0.3547E + 15

Acknowledgements. Dr. Mattheiss's research was supported in part by the office of Research and Projects at Southern Illinois University. Dr. Rubin's research was supported in part by grants from the Business Foundation of North Carolina. The authors are grateful to Dr. Brian K. Schmidt for pointing out several references and to Mr. Wayne Williksen for his invaluable programming efforts. The final version of this paper has benefitted from the perceptive comments made on an earlier draft by Dr. K. G. Murty and an anonymous referee.

References

[1] Balinski, M. L. (1961). An Algorithm for Finding all Vertices of Convex Polyhedral Sets. *SIAM* **IX** 72–78.

[2] Barnette, D. W. (1966). Trees in Polyhedral Graphs. *Canad. J. Math.* **XVIII** 731–736.

[3] ———. (1971). The Minimum Number of Vertices of a Simple Polytope. *Israel J. Math.* **X** 121–125.

[4] Brown, T. A. (1960). Hamiltonian Paths on Convex Polyhedra. Report P-2069, The Rand Corporation, Santa Monica, California.

[5] Burdet, C.-A. (1974). Generating All the Faces of a Polyhedron. *SIAM J. Appl. Math.* **XXVI** 479–489.

[6] Bürger, E. (1956). Über homogene lineare Ungleichungssysteme, *Z. Angew. Math. Mech.* **XXXVI** 135–139.

[7] Carrillo, M. (November 1977). Balinski's Enumeration Algorithm Revisited. Paper presented at ORSA/TIMS Joint National Meeting.

[8] Charnes, A., Cooper, W. W. and Henderson, A. (1953). *An Introduction to Linear Programming.* Wiley, New York.

[9] ——— and ———. (1961). *Management Models and Industrial Applications of Linear Programming*, vol. 1. Wiley, New York.

[10] Chernikov, S. N. (1968). *Linear Inequalities* (in Russian). Nauka, Moscow.

[11] Chernikova, N. V. (1964). Algorithm for Finding a General Formula for the Nonnegative Solutions of a System of Linear Equations. *U.S.S.R. Computational Mathematics and Mathematical Physics* **IV** 151–156.

[12] ———. (1965). Algorithm for Finding a General Formula for the Nonnegative Solutions of a System of Linear Inequalities. *U.S.S.R. Computational Mathematics and Mathematical Physics* **V** 228–233.

[13] Dahl, G. and Storøy, S. (October 1973). Enumeration of Vertices in the Linear Programming Problem. Report No. 45, University of Bergen.

[14] Dantzig, G. B. and Eaves, B. C. (1973). Fourier-Motzkin Elimination and its Dual. *J. Combinatorial Theory Ser. A* **XIV** 228–297.

[15] Duesing, E. C. (1977). *Polyhedral Convex Sets and the Economic Analysis of Production*. Unpublished Ph.D. dissertation, Department of Economics, University of North Carolina at Chapel Hill.

[15a] Dunham, J. R., Kelly, D. G. and Tolle, J. W. (December 1977). Some Experimental Results Concerning the Expected Number of Pivots for Solving Randomly Generated Linear Programs. Technical Report 77-16, Curriculum in Operations Research, University of North Carolina at Chapel Hill.

[16] Duffin, R. J. (1974). On Fourier's Analysis of Linear Inequality Systems. *Mathematical Programming Study 1*. American Elsevier Publishing Company, New York.

[17] Dyer, M. E. and Proll, L. G. (1977). An Algorithm for Determining All Extreme Points of a Convex Polytope. *Math. Programming* **XII** 81–96.

[18] ——— and ———. (16 February 1977). Letter to M. L. Balinski.

[19] ——— and ———. Vertex Enumeration in Convex Polyhedra—a Comparative Computational Study. Paper presented at CP77 Conference on Combinatorial Programming.

[20] ——— and ———. (July 1978). Personal Communication.

[21] Fourier, J. B. J. (1890). Solution d'une Question Particulière du Calcul des Inégalitiés. In *Oeuvres* **II** 317–328. Gauthier-Villars, Paris.

[22] Gal, T. and Nedoma, J. (1972). Multiparametric Linear Programming. *Management Sci.* **18** 406–422.

[23] Gale, D. (1963). Neighborly and Cyclic Polytopes. In *Proceedings of Symposia in Pure Mathematics* **VII** V. L. Klee, ed. American Mathematical Society, Providence, Rhode Island.

[24] Galperin, A. M. (1976). The General Solution of a Finite System of Linear Inequalities. *Math. Oper. Res.* **1** 185–196.

[25] Gerstenhaber, M. (1951). Theory of Convex Polyhedral Cones. In *Activity Analysis of Production and Allocation*, T. C. Koopmans, ed. Wiley, New York.

[26] Gomory, R. E. (1963). An Algorithm for Integer Solutions to Linear Programs. In *Recent Advances in Mathematical Programming*, R. L. Graves and P. Wolfe, eds. McGraw-Hill, New York.

[27] Greenberg, H. (1975). An Algorithm for Determining Redundant Inequalities and All Solutions to Convex Polyhedra. *Numer. Math.* **XXIV** 19–26.

[28] Grünbaum, B. (1967). *Convex Polytopes*. Wiley, New York.

[29] Hadley, G. (1962). *Linear Programming*. Addison-Wesley, Reading, Massachusetts.

[30] Klee, V. (1964). On the Number of Vertices of a Convex Polytope. *Canad. J. Math.* **XVI** 701–720.

[31] ———. (1974). Polytope Pairs and Their Relationship to Linear Programming. *Acta Math.* **CXXXI** 1–25.

[32] Kohler, D. A. (August 1967). Projections of Convex Polyhedral Sets. Report ORC 67-29, Operations Research Center, University of California at Berkeley.

[33] König, D. (1950). *Theorie der Endlichen und Unendlichen Graphen*. Chelsea, New York.

[34] Kuznetsov, V. G. (1966). Algorithms for Finding the General Solution of a System of Linear Inequalities. *USSR Computational Mathematics and Mathematical Physics* **VI** 197–205.

[34a] Liebling, T. M. (1973). On the Number of Iterations of the Simplex Method. In *Funfte Oberwohlfach-Tagung über Operations Research, Teil 2*. R. Henn, H. P. Künzi, and H. Schubert eds. Verlag Anton Hain, Meisenheim am Glan.

[35] Luenberger, D. G. (1973). *Introduction to Linear and Nonlinear Programming*. Addison-Wesley, Reading, Massachusetts.

[36] Manas, M. and Nedoma, J. (1968). Finding All Vertices of a Convex Polyhedron. *Numer. Math.* **XII** 226–229.

[37] Mattheiss, T. H. (1973). An Algorithm for Determining Irrelevant Constraints and All Vertices in Systems of Linear Inequalities. *Operations Res.* **21** 247–260.

[37a] ——— and Schmidt, B. K. (1977). The Probability that a Random Polytope is Bounded. *Math. Oper. Res.* **2** 292–296.

[38] ——— and Rubin, D. S. (September 1977). Comments on Dyer and Proll's Vertex Generating Algorithm. Technical Report 77-11, Curriculum in Operations Research, University of North Carolina at Chapel Hill.

[39] McMullen. P. (1970). The Maximum Number of Faces of a Convex Polytope. *Mathematika* **XVII** 179–184.

[40] Motzkin, T. S. (1936). Beitrage zur Theorie der Linearen Ungleichungen. Doctoral thesis, University of Zurich.

[41] ———, Raiffa, H., Thompson, G. L. and Thrall, R. M. (1953). The Double Description Method. In *Contributions to the Theory of Games*, II. H. W. Kuhn and A. W. Tucker, eds. Annals of Mathmematics Study, No. 28, Princeton University Press, Princeton, New Jersey.

[42] Murty, K. G. (1968). Solving the Fixed Charge Problem by Ranking the Extreme Points. *Operations Res.* **XVI** 268–279.

[42a] ———. (1971). Adjacency on Convex Polyhedra. *SIAM Rev.* **XIII** 377–386.

[43] Pollatschek M. and Avi-Itzhak, B. (1969). Sorting Extremem Point Solutions of a Linear Program. Paper presented at Third Annual Israel Conference on Operations Research.

[44] Remez, E. Ya. and Shteinberg, A. S. (1967). A Theorem of Convex Polyhedra in Connection with the Problem of Finding the Set of Solutions to a System of Linear Inequalities. *Ukrainian Math. J.* **XIX** 191–202.

[45] Rosenfeld, M. and Barnette, D. Hamiltonian Circuits in Certain Prisms. Undated mimeo, Mathematics Department, University of California at Davis.

[46] Rubin, D. S. (1972). Redundant Constraints and Extraneous Variables in Integer Programs. *Management Sci.* **18** 423–427.

[47] Schmidt, B. K. and Mattheiss, T. H. (October 1975). On the Expected Value of the Number of Vertices of a Convex Polytope. Paper presented at ORSA/TIMS Joint National Meeting.

[48] Shachtman, R. H. (1974). Generation of the Admissible Boundary of a Convex Polytope. *Operations Res.* **22** 151–159.

[49] Shefi, A. (1969). *Reduction of Linear Inequality Constraints and Determination of All Feasible Extreme Points*. Unpublished Ph.D. dissertation, Department of Engineering—Economic Systems, Stanford University. (See discussion in [31, Chapter 5].)

[50] Sherman, B. F. (1977). A Counterexample to Greenberg's Algorithm for Solving Linear Inequalities. *Numer. Math.* **XXVII** 491–492.

[51] Silverman, G. J. (June 1971). Computational Considerations in Extreme Point Enumeration. Report G 320-2649, IBM Los Angeles Scientific Center.

[52] Thompson, G. L., Tonge, F. M. and Zionts, S. (1966). Techniques for Removing Nonbinding Constraints and Extraneous Variables from Linear Programming Problems. *Management Sci.* **12** 588–608.

[53] Tucker, A. W. (January 1958). Condensed Schemata for Dantzig's Simplex Method. Mimeographed notes, Princeton University.

[54] Tutte, W. T. (1946). On Hamiltonian Circuits. *J. London Math. Soc.* **XXI** 98–102.

[55] Uzawa, H. (1958). A Theorem on Convex Polyhedral Cones. In *Studies in Linear and Non-Linear Programming*, K. J. Arrow, L. Hurwicz, and H. Uzawa, eds. Stanford University Press, Stanford, California.

[56] Walker, M. R. (1973). Determination of the Convex Hull of a Finite Set of Points. Unpublished M.S. thesis, Curriculum in Operations Research, University of North Carolina at Chapel Hill.

[57] Wets, R. J.-B. and Witzgall, C. (1966). Algorithms for Frames and Lineality Spaces of Cones. *J. Res. Nat. Bur. Standards Sect. B* **71** 1–7.

MATTHEISS: COLLEGE OF COMMERCE AND BUSINESS ADMINISTRATION, UNIVERSITY OF ALABAMA, TUSCALOOSA, ALABAMA

RUBIN: CURRICULUM IN OPERATIONS RESEARCH AND SCHOOL OF BUSINESS ADMINISTRATION, UNIVERSITY OF NORTH CAROLINA, CHAPEL HILL, NORTH CAROLINA 27514

. . . etc., H_i in a nested fashion until some 2-face of the polytope is indexed, certai whose vertices are obtained by successive iterations of the simplex method. When of the 2-faces belonging to H_i have been processed the algorithm drops that half-sp requirement, chooses another defining hyperplane, say H_j, and proceeds to find vertices of the polytope lying in H_j which are not in H_i. The procedure continues there are n Half-space requirements remaining, and the cone they define determ the last vertex. If the last vertex in an $(n - 1)$-face to be listed is not adjacent to a the unlisted vertices, a backtracking procedure comprising an unspecified sequenc pivots is necessary to arrive at an unlisted vertex. It is important to note that w half-space requirements are dropped, the algorithm is allowed to visit vertex poin the polyhedral set defined by the reduced set of inequalities which may not be vert of the original polytope. The primary advantage of Balinski's algorithm is its mod storage requirement. All that needs be in core storage is the current tableau and a l index sets. A sketch of Balinski's algorithm is given in Appendix A. Carrillo [7] recently discussed an implementation of Balinski's algorithm based on the simplex method.

Silverman [51] gives the following summary of Murty's algorithm which published in 1968. The method of Murty [42] is designed to solve the fixed char problem by ranking the extreme points in nondecreasing order of a linear objec function. The method is based on the intuitively appealing result that if we have a of vertices, $v_1, v_2, \ldots v_{k-1}$, ranked in nondecreasing order by some objective vec c, then the next element in the sequence, v_k, must be adjacent to one of the vert already enumerated. The general step of the method beings with $v_1, v_2, \ldots,$ already recorded along with pivot elements and objective value for each cost non creasing extreme point adjacent to v_j for $j = 1, 2, \ldots, k - 1$. By the above result v_k adjacent to one of the previous v_j and a comparison of objective values for smallest value greater than or equal to v_{k-1} will yield v_k. Then all of its adjac vertices must be recorded in order to determine v_{k+1}. If a listed extreme point degenerate then all extreme points adjacent to v_k may not be reached from the table for v_k by one pivot. In this case all feasible bases that represent v_k must be sto along with all extreme points adjacent to each one. This causes every adjacent b feasible solution to be recorded.

Murty [42, p. 277–278] discusses how his method might be implemented o computer. It is clear that the number of pivots will be equal to the number of ba feasible solutions less one, which is the optimal extreme point of the linear progr Thus the only computational considerations involve the storage organization. Mu suggests three arrays:

Array 1. All the objective values of the basic feasible solutions adjacent to ran extreme points.

Array 2. All basic solutions that have already been ranked.

Array 3. The basic feasible solutions corresponding to the objective values sto in Array 1.

Murty suggests locating Arrays 1 and 2 in core and Array 3 on tape. The o problem is that not enough information may be available in these arrays to guara successful enumeration of all extreme points. If we have just determined v_{k-1}, presumably have the tableau for v_{k-1} in core, then the information in Arrays 1 a is sufficient to determine the basic feasible solution v_k and its objective value. If not adjacent to v_{k-1} then pivot operations on the tableau of v_k are necessary determine all its adjacent cost nondecreasing extreme points for storage in Array and 3. Since v_k may be adjacent to any one of $v_1, v_2, \ldots, v_{k-1}$ all these table must be available. If these tableaux are stored on tape and selected by the algorit in a random fashion, a great deal of time will be spent in tape access. A far m

ficient implementation would have the tableaux stored on a high speed direct access vice such as drum or disk. If enough high speed direct access storage is available, e method of Murty may be most efficient for a nondegenerate problem because vot operations will be limited by the number of extreme points. The amount of put and output of tableaux is at least one tableau output per extreme point and one leau input each time v_j is not adjacent to v_{j-1}. Thus computational efficiency pends on the problem size, the path of the algorithm and the computer configura-

Curiously enough, what might be considered the most direct approach was not blished until 1968 when Manas and Nedoma [36] gave their algorithm. The gorithm has a hyperplane oriented bookkeeping system and a simplex tabular presentation. A complete statement of the algorithm is given in Appendix B.

Computationally this algorithm involves the efficient management of a list of creasing size. The number of elements on the list will grow until it equals the mber of vertices of the polytope. At least one simplex tableau must be analyzed for ery vertex of the polytope. A simplex tabular representation and the possibility of ternal list maintenance and manipulation appear to make this algorithm feasible for rger problems, although processing time may become unreasonably long due to avy list searching requirements. This algorithm has been employed by Gal and edoma [22] at the core of their Multiparametric Linear Programming procedures, hough any other technique described here would also be viable in that context.

The ranking method of Pollatschek and Avi-Itzhak [43] begins with the vertex v_1, termined as the optimal solution to the linear program of minimizing cx on the lyhedron. The extreme point adjacent to v_1 which has the lowest value of cx is lected as v_2, and the constraint to $cx < cv_2$ is added to the system. This introduces n rtificial basic solutions" in the nondegenerate case; these "artificial basic solutions" e not extreme points of the original polyhedron. The method proceeds to pivot ound the new constraint (similarly to Balinski's method) noting the objective values the adjacent extreme points. The one with the smallest objective value becomes v_3; e constraint $cx > cv_3$ is added; and the process is repeated. Suppose $v_1, \ldots, v_j$ have en ranked. In general, the number of "artificial basic solutions" on the hyperplane $= cv_j$ is equal to the number of pairs (v_r, v) where v_r is an already ranked extreme int with $cv_r < cv_j$, and v is an adjacent extreme point of v_r on the original lyhedron with $cv > cv_j$. (See Murty [42a].) This can be a very large number, even in e nondegenerate case. Any one of these (v_r, v) pairs appears repeatedly in several ps of the algorithm as an "artificial basic solution" on the hyperplane $cx = cv_j$ until e vertex v joins the ranked sequence. Thus, while the Pollatschek/Avi-Itzhak ethod avoids the out of core storage requirements of the Murty method, it reconucts the information stored in Murty's Array 1 repeatedly after each ranking step. is fact, in addition to complicated bookkeeping requirements renders this approach fficult to program and computationally inferior to the Murty algorithm.

Silverman [51] defines a modification of the Hamiltonian path, called a G-path. A path corresponds to a sequence of pivots such that each vertex of the polytope is on adjacent to a vertex on the path. There is no assurance of the existence of a G-path, s the necessity of a backtracking procedure in this algorithm. The actual path lowed by the Manas and Nedoma algorithm would seem to be often a G-path. e computer time for record keeping operations is required by Silverman's method n for the other pivoting methods, as he so states.

The method of Dahl and Storøy [13] ranks all vertices $v_1, v_2, \ldots, v_p$ (corresponding $x_1, x_2, \ldots, x_p$, the solutions of a linear program, having objective vector c) in a quence such that $cx_1 > cx_2 > \cdots cx_p$. This algorithm can be stated as the following dification of the Manas and Nedoma algorithm